Peekaboom: A Game for Locating Objects in Images

Luis von Ahn, Ruoran Liu and Manuel Blum
Computer Science Department, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh PA 15213
{biglou, royliu, mblum}@cs.cmu.edu

ABSTRACT
We introduce Peekaboom, an entertaining web-based game that can help computers locate objects in images. People play the game because of its entertainment value, and as a side effect of them playing, we collect valuable image metadata, such as which pixels belong to which object in the image. The collected data could be applied towards constructing more accurate computer vision algorithms, which require massive amounts of training and testing data not currently available. Peekaboom has been played by thousands of people, some of whom have spent over 12 hours a day playing, and thus far has generated millions of data points. In addition to its purely utilitarian aspect, Peekaboom is an example of a new, emerging class of games, which not only bring people together for leisure purposes, but also exist to improve artificial intelligence. Such games appeal to a general audience, while providing answers to problems that computers cannot yet solve.

Author Keywords
Distributed knowledge acquisition, object segmentation, object recognition, computer vision, Web-based games.

ACM Classification Keywords:
I.2.6 [**Learning**]: Knowledge acquisition. H.5.3 [**HCI**]: Web-based interaction.

INTRODUCTION
Humans understand and analyze everyday images with little effort: what objects are in the image, where they are located, what is the background, what is the foreground, etc. Computers, on the other hand, still have trouble with such basic visual tasks as reading distorted text or finding where in the image a simple object is located. Although researchers have proposed and tested many impressive algorithms for computer vision, none have been made to work reliably *and* generally.

Most of the best approaches for computer vision (e.g. [4,5,9,10]) rely on machine learning: train an algorithm to perform a visual task by showing it example images in which the task has already been performed. For example, training an algorithm for testing whether an image contains a dog would involve presenting it with multiple images of dogs, each annotated with the precise location of the dog in the image. After processing enough images, the algorithm learns to find dogs in arbitrary images. A major problem with this approach, however, is the lack of training data, which, obviously, must be prepared by hand. Databases for training computer vision algorithms currently have hundreds or at best a few thousand images [13] — orders of magnitude less than what is required.

In this paper we address the problem of constructing a massively large database for training computer vision algorithms. The target database will contain millions of images, all fully annotated with information about what objects are in the image, where each object is located, and how much of the image is necessary to recognize it. Our database will be similar to those previously shown to be useful for training computer vision algorithms (e.g. [13]).

To construct such a database, we follow the approach taken by the ESP Game [1] and introduce a new game called Peekaboom. Peekaboom is an extremely enjoyable networked game in which, simply by playing, people help construct a database for training computer vision algorithms. We guarantee the database's correctness even if the people playing the game don't intend it. As we will show in this paper, our game is also very enjoyable, with some people having played over 40 hours a week. We will further show that this game can be used to improve image-search results and to calculate object bounding-boxes similar to those in Flickr [8] (see Figure 7).

The ESP Game [1] is an interactive system that allows people to label images while having fun. The ESP Game collects random images from the Web and outputs word labels describing the contents of the images. The game has already collected millions of labels for arbitrary images. Given an image, the ESP Game can be used to determine what objects are in the image, but cannot be used to determine where in the image each object is located. Such location information is necessary for training and testing computer vision algorithms, so the data collected by the ESP Game is not sufficient for our purposes. The game introduced in this paper, Peekaboom, improves on the data collected by the ESP Game and, for each object in the image, outputs precise location information, as well as other information useful for training computer vision algorithms. By playing a game, people help us collect data

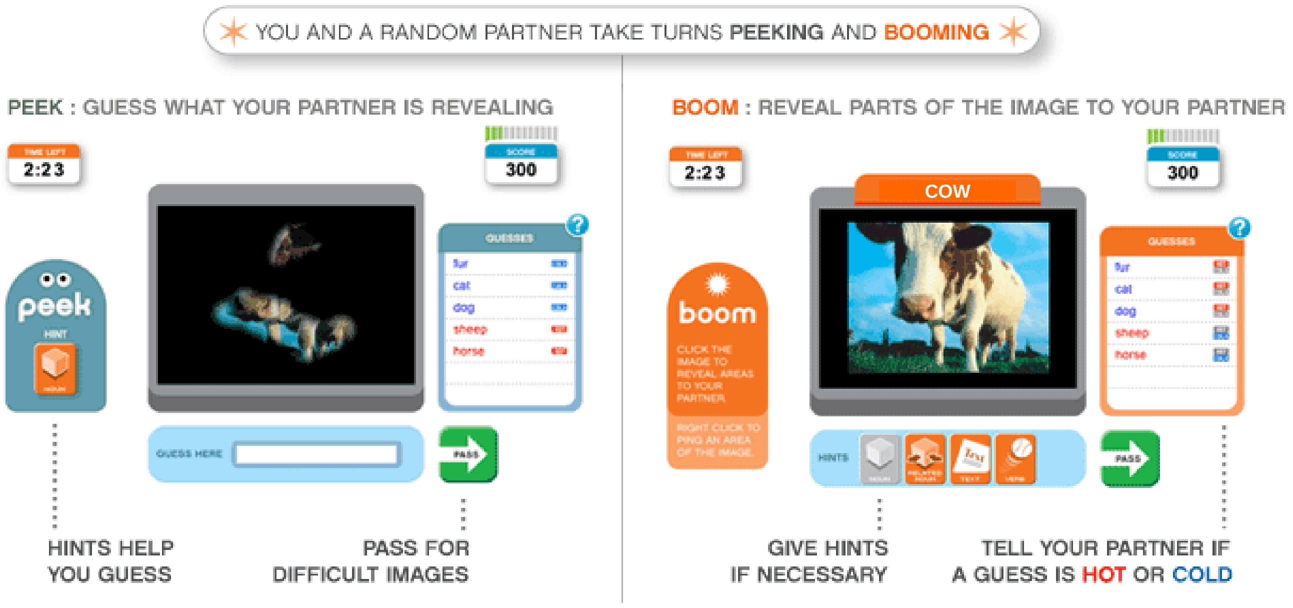

YOU AND A RANDOM PARTNER TAKE TURNS PEEKING AND BOOMING
PEEK : GUESS WHAT YOUR PARTNER IS REVEALING
BOOM : REVEAL PARTS OF THE IMAGE TO YOUR PARTNER
2:23
300
COW
peek
boom
fur
cat
dog
sheep
horse
PASS
HINTS HELP YOU GUESS
PASS FOR DIFFICULT IMAGES
GIVE HINTS IF NECESSARY
TELL YOUR PARTNER IF A GUESS IS HOT OR COLD

Figure 1. Peek and Boom. Boom gets an image along with a word related to it, and must reveal parts of the image for Peek to guess the correct word. Peek can enter multiple guesses that Boom can see.

not because they want to be helpful, but because they have fun. Indeed Peekaboom (or the ESP Game or any game built on this premise) can be treated as a "human algorithm": on input an image, it outputs (with arbitrarily high probability) a correct annotation of the image. Instead of using a silicon processor, this "algorithm" runs on a processor consisting of regular humans interacting throughout the Web.

In addition to applications in computer vision and image search, our system makes a significant contribution to HCI because of the way it addresses the problem: Peekaboom presents an example of a new line of research devoted to solving large-scale problems with human computing power, where people *interact* with computers to extend the computational abilities of machines.

BASIC GAME PLAY

Peekaboom, as the name may suggest, is a game with two main components: "Peek" and "Boom." Two random players from the Web participate by taking different roles in the game — when one player is Peek, the other is Boom. Peek starts out with a blank screen, while Boom starts with an image and a word related to it (see Figure 1).

The goal of the game is for Boom to reveal parts of the image to Peek, so that Peek can guess the associated word. Boom reveals circular areas of the image by clicking. A click reveals an area with a 20-pixel radius. Peek, on the other hand, can enter guesses of what Boom's word is. Boom can see Peek's guesses and can indicate whether they are hot or cold.

When Peek correctly guesses the word, the players get points and switch roles; play then proceeds on a new image-word pair. If the image-word pair is too difficult, the two players can "pass," or opt out, of the current image. Passing creates the same effect as a correct guess from Peek, except that the players get no points.

To maximize points, Boom has an incentive to reveal only the areas of the image necessary for Peek to guess the correct word. For example, if the image contains a car and a dog and the word associated to the image is "dog," then Boom will reveal only those parts of the image that contain the dog. Thus, given an image-word pair, data from the game yield the area of the image pertaining to the word.

Pings

Another component of the game are "pings" — ripples that appear on Peek's screen when Boom right-clicks on the image (see Figure 2). If two players were playing with the image on Figure 2, then many correct words are possible from Peek's point of view: elephant, trunk, tusk, ear. Suppose the correct word is "trunk." To get Peek to guess correctly, Boom can "ping" the trunk of the elephant by right-clicking on it. In doing so, Boom helps to disambiguate the trunk from the rest of the elephant.

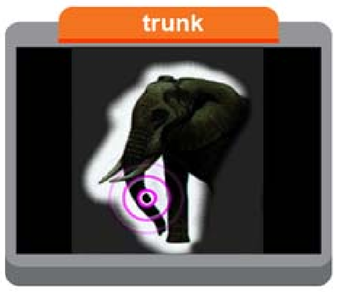


Figure 2. Pings. To help Peek, Boom can "ping" parts of the image by right-clicking on them.

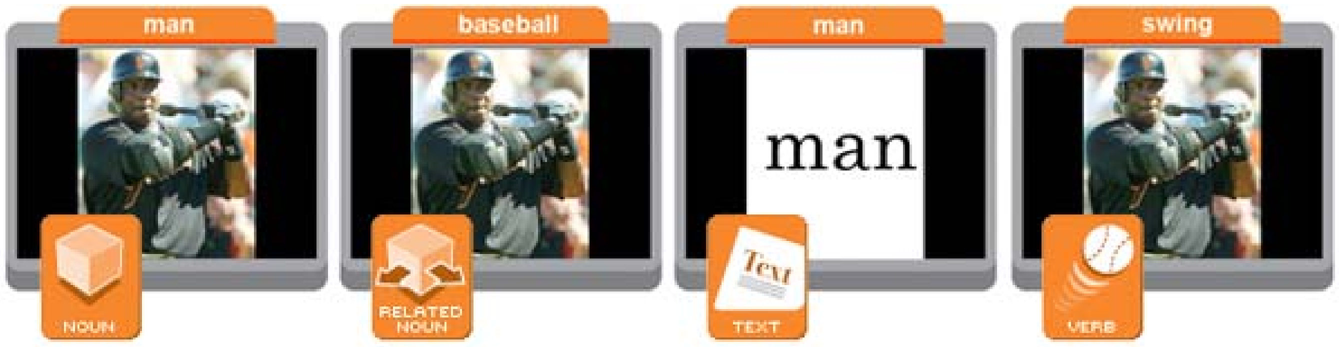


Figure 3. Hints. Boom can further help Peek by giving hints about how the word relates to the image: is it a noun describing something in the image, a noun related to the image, text on the image, or a verb?

Hints

Another feature of the game are buttons that allow Boom to give hints to Peek about how the word relates to the image (See Figures 1 and 3). Upon Boom's pressing of one of the hint buttons, a corresponding flashing placard appears on Peek's screen. The reason for having hints is that often the words can relate to the image in multiple ways: as nouns, verbs, text, or related nouns (something not in the image, but related to it).

THE ORIGIN OF IMAGES AND LABELS

All words presented to the players are related to their corresponding image. On input an image-word pair, Peekaboom outputs a region of the image that is related to the word. We obtain millions of images with associated keyword labels from the ESP Game [1], which we now describe in more detail.

As mentioned before, the ESP Game is a two-player online game that pairs random players from the Web. From the player's perspective, the goal of the ESP Game is to guess the word that their partner is typing for each image. Once both players have typed the same string, they move on to a next image. Since the players can't communicate and don't know anything about each other, the easiest way for both to type the same string is by typing something related to the common image. The string upon which the two players agree is a very good label for the image. We use the labels collected from the ESP Game as the words we present to the players in Peekaboom.

GAME POINTS AND THE BONUS ROUND

Although the exact number of points given to the players for different actions is not important, we mention it to show the relative proportions. Furthermore, we mention the different point strategies used by Peekaboom to keep players engaged.

Points are given to both Peek and Boom equally whenever Peek guesses the correct word. In the current implementation, both obtain 50 points. Points are not subtracted for passing. Points are also given to both Peek and Boom for using the hint buttons. Although this might appear counterintuitive since using hints deducts points in many other games, we actually *want* the players to use the hint buttons. As mentioned above, hints give us additional information about the relationship between the word and the image, and therefore we encourage players to use them. Twenty-five extra points are given to both Peek and Boom whenever Peek guesses the correct word and Boom had used a hint. Points are not given for usage of the hot/cold buttons.

Every time the players correctly complete four images, they are sent to a "bonus" round. The bonus round is different in nature from the rest of the game and allows players to obtain up to 150 points. In the bonus round (see Figure 4), players simply click on an object in the image. The closer they are to each other's clicks, the more points they get. For example, both players could obtain an image of a car and be told "click on the car."

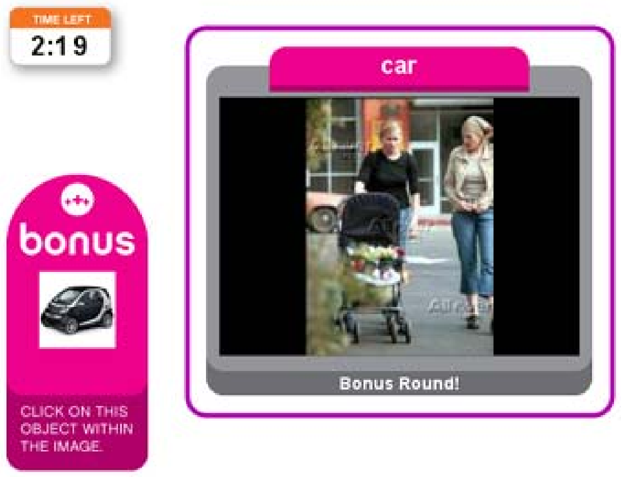


Figure 4. The Peekaboom Bonus Round. Players must click on the specified object within the image; they obtain points proportional to how close their clicks are to each other's clicks.

Players obtain between 0 and 10 points for every click in the bonus round, depending on how far the click is from their partner's corresponding click. The bonus round is timed: players have to click on the same place as their partner as many times as they can in 5 seconds. If the object is not in the image, players can pass. Because some images do not contain the object related to the word, passing in the bonus round generates 25 points for both players (so we can learn whether the object is there). Players cannot pass after they have clicked on the image.

There are two reasons for the Peekaboom bonus round. First, by giving players "bite-size" milestones (getting four images correctly), we reinforce their incremental success in the game and thus encourage them to continue playing. Second, the bonus round is an alternative approach to collecting training data for computer vision. In it, players click inside specific objects within an image. Such clicks give additional information for training computer vision algorithms. In this paper we do not concern ourselves with such information, but remark that it is also useful.

COLLECTING IMAGE METADATA

Our goal is to construct a database for training computer vision algorithms. Here we discuss exactly what information is collected by Peekaboom and how it is collected.

On input an image-word pair (coming directly from the ESP Game), Peekaboom collects the following information:

- **How the word relates to the image.** Is it an object, person, or animal in the image, is it text in the image, is it a verb describing an action in the image, is it an object, person, or animal not in the image but related to it? The ESP Game associates words to images, but does not say how the word is related to the image. Figure 3, for instance, shows multiple ways in which a word can be related to an image. Hint buttons in Peekaboom allow us to determine the relation of the word to the image. This is useful in multiple ways, but for the purposes of constructing training sets for computer vision, it allows us to weed out "related nouns" and to treat "text" separately.
- **Pixels necessary to guess the word.** When Peek enters the correct word, the area that Boom has revealed is precisely enough to guess the word. That is, we can learn exactly what context is necessary to determine what the word refers to. This context information is absolutely necessary when attempting to determine what type of object a set of pixels constitutes (see Figure 5).
- **The pixels inside the object, animal, or person.** If the word is a noun directly referring to something in the image, "pings" give us pixels that are inside the object, person, or animal.

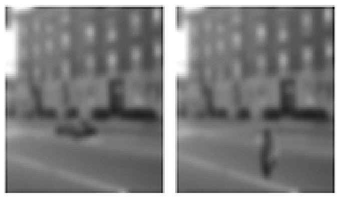


Figure 5. The image on the left contains a car driving through the street, while the one on the right has a person crossing the same street. Both the car and the person are exactly the same set of pixels up to a rotation by 90 degrees. (Example taken from [11].)

- **The most salient aspects of the objects in the image.** By inspecting the sequence of Boom's clicks, we gain information about what parts of the image are salient with respect to the word. Boom typically reveals the most salient parts of the image first (e.g., face of a dog instead of the legs, etc.).
- **Elimination of poor image-word pairs.** If many independent pairs of players agree to pass on an image without taking action on it, then likely they found it impossibly hard because of poor picture quality or a dubious relation between the image and its label. By implementing an eviction policy for images that we discover are "bad," we can improve the quality of the data collected (as well as the fun level of the game).

When multiple players have gone through the same image, these pieces of information can be combined intelligently to give extremely accurate and useful annotations for computer vision. Later in the paper, for example, we show how a simple algorithm can use the data produced by Peekaboom to calculate accurate object bounding-boxes (see Figure 7).

THE SINGLE PLAYER GAME

Peekaboom is a two-player game. Oftentimes, however, there will be an odd number of people attempting to play the game, so the remaining person cannot be paired. To prevent their frustration, we also have a single-player version of the game in which the player is matched with a server-side "bot."

Our bot acts intelligently to simulate a human player by being based on pre-recorded games. In other words, we take data collected from pairs of humans and use it as the basis for the computer player's logic. Emulating a Boom player is fairly simple: the bot can regurgitate the sequence of recorded clicks to the human. Emulating Peek is much more complicated; the bot needs to have some concept of closeness of the human's clicks to the set of recorded clicks. For instance, if the human does not reveal the dog in the picture, the bot should not guess "dog." Our bot only reveals a certain pre-recorded guess if enough area has been revealed. Towards this end, it employs a spatial data structure whose members are circles, each of which corresponds to a click. Elements of the data structure are removed as they are clicked on by the human player. When the data structure becomes empty, the bot gives the correct answer. Moreover, it has the ability to make incorrect guesses along the way, based on the relative emptiness of the spatial data structure.

CHEATING

Peekaboom is a collaborative game: partners work together to maximize their score. When both partners do not communicate outside the game environment, we obtain correct information. However, if the two partners collude to cheat on the game, the data could be poisoned. For instance, if Boom and Peek know each other and have an outside means of communication, then Boom can simply tell Peek what words to type.

Peekaboom contains multiple anti-cheating mechanisms. Through a combination of online in-game enforcement and offline analysis, we are able to detect and deal with cheating. Before detailing Peekaboom's anti-cheating measures, we mention that cheating attempts are uncommon. Although a minority of players might obtain satisfaction from "gaming the system," the majority of them just want to play the game honestly. Indeed, as anecdotal evidence, when Peekaboom was tested in a room with children of ages 9-13, they would cover the word with their hand to prevent others in the room from seeing the answers. Nevertheless, Peekaboom does have a full set of measures to prevent collusion.

- **The player queue.** When players log on to the game server, they are not immediately paired off. Instead, the server makes them wait *n* seconds, where *n* is the number of seconds until the next matching interval. Currently, matching intervals happen every 10 seconds, and when they do, the server matches everyone in the queue with a partner (any odd person out will be paired with a bot). With a large number of players in the system, we can ensure that a player's partner is random and prevent colluders from getting matched just because they clicked "start playing" at the same time.
- **IP address checks.** We also check player's IP addresses to ensure that they are not paired with themselves or with others that have a similar address (similarity in IP addresses can imply geographical proximity).
- **Seed images.** Because our system is a web-based game, one point of concern is that bots (i.e. automated players) might play the game and pollute the pool of collected data. To detect them, we introduce seed images into the system; in other words, those for which we have hand-verified metadata. On being presented seed images, if a player consistently fails to click on the relevant parts when playing Boom or to guess the correct words when playing Peek, they will be added to a blacklist. We discard all current and future game play data associated with anyone on the blacklist. Notice that, almost by definition, a computer program cannot successfully play Peekaboom — if it were able to do so, then it would be able to recognize the objects in the images. Therefore, this strategy prevents bots (as well as otherwise malicious players) from poisoning our data.
- **Limited freedom to enter guesses.** Since Boom can see all of Peek's guesses, the game allows a limited form of communication between the players. Indeed, many of the Peekaboom players use the guess field as a way to communicate with their partner. It is not uncommon for the first guess in a game to be "hi" or for the first guess after passing on an image to be the correct word associated with the previous image. It is also not uncommon for players to type "sorry" after taking too long on an image. A possible cheating strategy is to exchange IM screen names through the guess field and then, using IM, communicate the correct words. Although we have never observed attempts to execute such a strategy, we can mitigate it by not allowing Peek to enter any non-alphabetical characters (such as numbers). Similarly, we can prevent Boom from seeing any guesses that are not words in the dictionary (currently we do allow Boom to see such guesses because we have not seen players attempt to cheat in this way). However, even if players are successful in such a strategy, the other anti-collusion mechanisms can deal with the corrupted data.
- **Aggregating data from multiple players.** In addition to the above strategies, we aggregate data from multiple players for a given image-word pair. By doing this, we can eliminate outliers.

IMPLEMENTATION

We implemented the architecture of the game under the client-server model. The client application is delivered as a Java applet, while the server is written purely in Java. Applets connect to a server, http://www.peekaboom.org, which then matches the players with games of Peekaboom. Upon two players' completion of a match, the server writes their game play data and scores to disk. We then compile the collected data into desired formats.

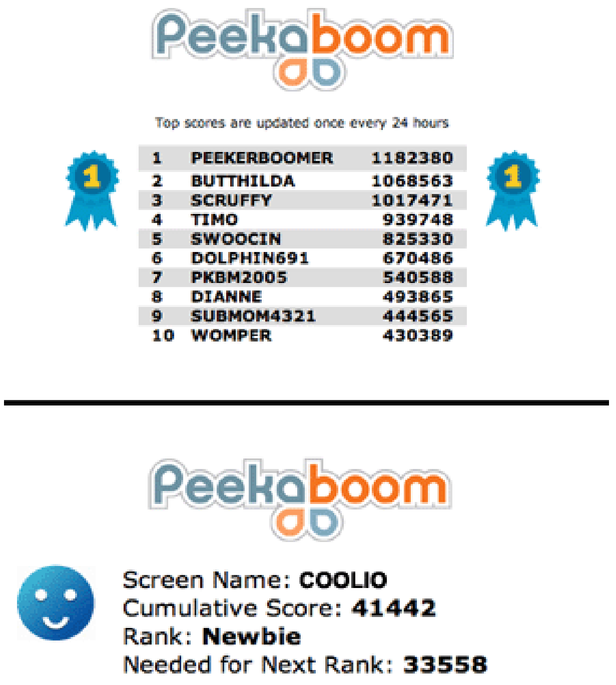
Our implementation of the game contains many features to improve game-play:

- **Spelling check.** Incorrectly spelled words are displayed in a different color to notify players. This is important because the Peek player usually types multiple guesses in a short time, often making spelling mistakes.

- **Inappropriate word replacement.** Since Boom can see Peek's guesses, we do not allow Peek to enter inappropriate words. Whenever one of Peek's guesses is among a list of possible inappropriate words, we substitute it with another word chosen from a list of innocent words such as "love," "caring," "ILuvPeekaboom," etc.

- **Top scores list and ranks.** The Peekaboom website prominently displays the cumulative top scores of the day as well as the top scores of all time. Furthermore, players are given a rank based on the total number of points they have accumulated throughout time (see Figure 6). The different ranks are: Fresh Meat (0-15,000 points), Newbie (15,000-75,000 points), Player (75,000-250,000 points), Gangster (250,000-1,000,000 points) and Godfather (1,000,000 or more points).

  We remark that ranks have proven an important component of Peekaboom's incentive strategy. Of the 15,000+ players that have obtained an account, 47% of them have scores that fall within 5,000 points of the rank cutoffs. Given that these intervals cover less than 2% of the space of possible cumulative scores, this strongly suggests that many players simply play to reach a new rank.



1	PEEKERBOOMER	1182380
2	BUTTHILDA	1068563
3	SCRUFFY	1017471
4	TIMO	939748
5	SWOOCIN	825330
6	DOLPHIN691	670486
7	PKBM2005	540588
8	DIANNE	493865
9	SUBMOM4321	444565
10	WOMPER	430389

Screen Name: **COOLIO**
Cumulative Score: **41442**
Rank: **Newbie**
Needed for Next Rank: **33558**

Figure 6. Top scores and player ranks. Players are shown their current rank and the number of points remaining for the next rank.

ADDITIONAL APPLICATIONS

Before going to the evaluation section, we mention two additional applications for the data collected by Peekaboom. A benefit of these applications is that they are "direct" in that they do not require the training of machine learning algorithms.

Improving Image-Search Results

Peekaboom gives an accurate estimate of the fraction of the image related to the word in question. This estimate can be calculated from the area revealed by Boom. The fraction of the image related to a word can be used to order image-search results: images in which the word refers to a higher fraction of the total pixels should be ranked higher. Much like the goal of the ESP Game is to label all images on the Web, we can imagine Peekaboom doing the same and thus further improving image-search.

Object Bounding-Boxes

In the same vein, Peekaboom can be used to directly calculate object bounding-boxes similar to those used in Flickr [8] (see Figure 7). Flickr is a photo sharing service that allows users to "tag" images with keywords and to associate keywords with rectangular areas in the image (the areas and tags, however, are not guaranteed to be correct since a user can enter anything they wish for their own images). To exhibit the power of the data collected by Peekaboom, we show how to use it calculate such rectangles. We emphasize, however, that the data collected by Peekaboom is significantly richer and that to calculate the rectangles, we discard vast amounts of the information collected by our game.

Since Peekaboom annotates arbitrary images on the Web, its data allows for an image search engine in which the results are highlighted (similar to the highlighted words in Google's text search results). Using the data obtained in the first two weeks of game-play, we have implemented a prototype of such a search engine (see Figure 7). The search engine can be accessed from the Peekaboom website: http://www.peekaboom.org.

The bounding boxes were calculated as follows. For a single play of an image-word pair, we create a matrix of 0s and 1s. The dimensions of the matrix are the same as the dimensions of the image (in pixels). At first, every entry in the matrix is a 0. We add a 1 in every pixel clicked by Boom, as well as in the circle of radius 20 pixels around the click. We thus obtain a matrix of 0s and 1s corresponding to the exact area that was revealed in a single game-play. Next, we combine different plays of the same image-word pair by adding their corresponding matrices. This gives a matrix whose entries are integers corresponding to the number of different players that revealed each pixel of the image. On this combined matrix, we apply a threshold of 2, meaning that we substitute every value less than 2 with 0 and every value greater than 2 with 2. This gives a matrix corresponding to all the pixels that have been revealed by at least 2 players. Next, we cluster these pixels and calculate

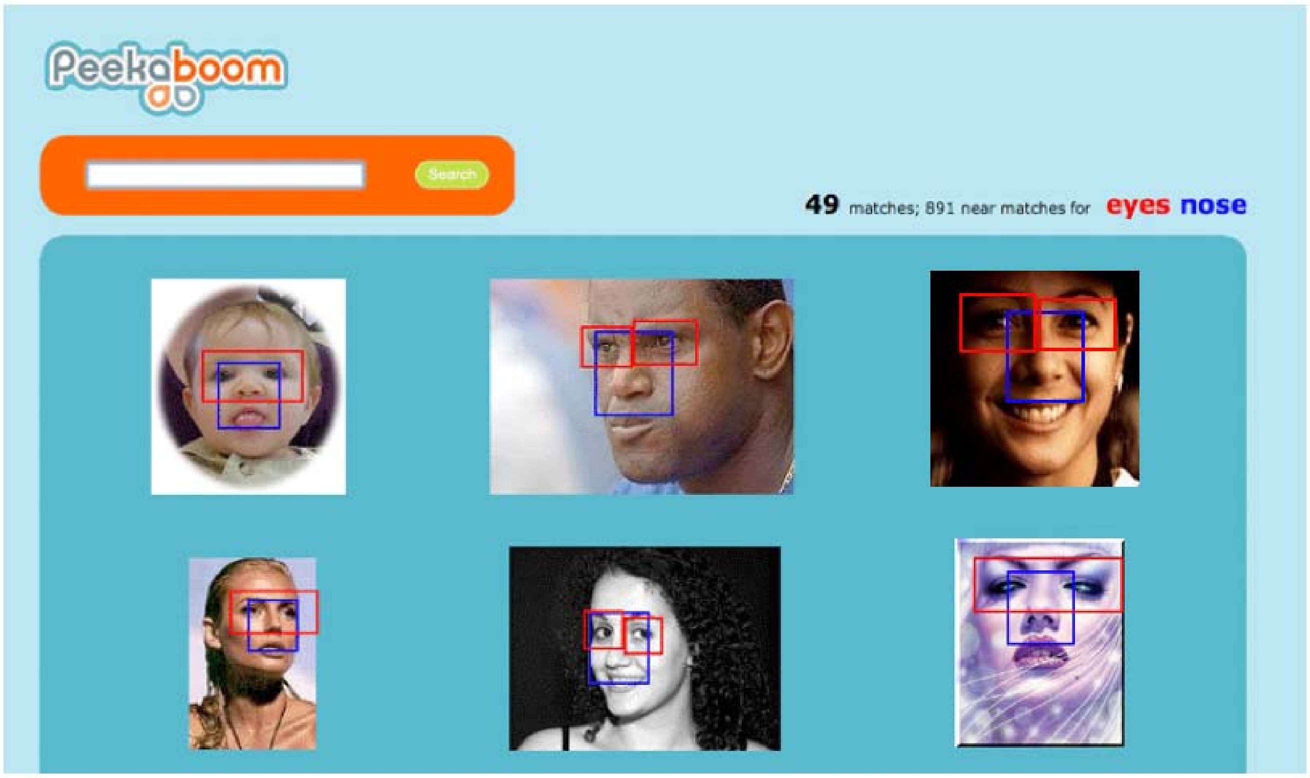


Figure 7. Object bounding-boxes obtained from Peekaboom data.

the bounding boxes by taking, for each cluster, the leftmost, rightmost, topmost and bottommost points. This algorithm may produce multiple bounding-boxes for a single image-word pair. For instance, in Figure 7, we can see that many of the results for "eyes" have two-bounding boxes, one corresponding to each eye.

As we will see, the results produced by this simplistic algorithm are extremely accurate. Such results could be improved by making intelligent use of the additional data given by Peekaboom (such as pings, the precise order of the areas revealed, etc.), but for the purposes of this paper, we use the simplistic algorithm.

Alternative: Using Ping Data for Pointing

Instead of showing bounding-boxes calculated from revealed areas, we could show arrows or lines pointing to the objects (see Figure 8). Such pointers can be easily calculated from the ping data. The simplest algorithm for doing so is to select a ping at random and assume it is a good pointer for the object. We will show that this simplistic algorithm gives very accurate results. (Figure 8 shows an image in which the different objects have been located using ping data.) More elaborate algorithms could give even better results. We remark, however, that simply "averaging" the pings over multiple players to obtain a single pointer does not give accurate results. For instance, if the object was "eyes," averaging the pings gives a pointer to a region that is not an eye.

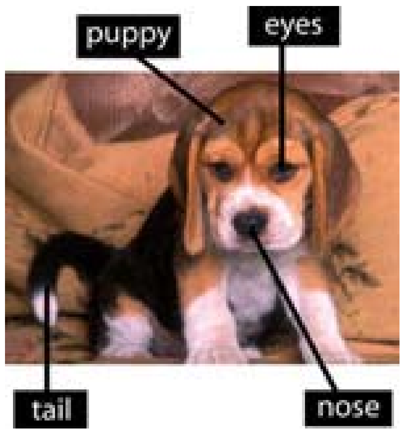


Figure 8. Calculation of object pointers using pings.

EVALUATION: USER STATISTICS

The evaluation of our claims consists of two parts. First, we must show that the game is indeed enjoyable. Second, we must show that the data produced by the game is accurate.

It is difficult to evaluate how enjoyable a game really is. One approach is to ask participants a series of questions regarding how much they enjoyed playing the game. Our data for such an approach were extremely positive, but we follow a different approach in this paper: we present usage statistics from arbitrary people playing our game online (this same approach was used by the ESP Game [1]).

Usage Statistics
Peekaboom was released to a general audience on August 1 of 2005. We present the usage statistics from the period starting August 1, 2005 and ending September 1, 2005. A total of 14,153 different people played the game during this time, generating 1,122,998 pieces of data. By "different people" we mean different user IDs. By a "piece of data," we mean a successful round of Peekaboom in which Peek correctly guessed the word given Boom's revealed region. We mention that an image-word pair can have multiple pieces of data associated to it if it occurs in multiple games.

If 14,153 people gave us 1,122,998 pieces of data, then on average each person played on 158.68 images. Since each session of the game lasts 4 minutes and on average players go through 8.7 images during a session, in this one month period each person played on average 72.96 minutes (without counting time spent waiting for a partner, etc.).

Over 90% of the people played on more than one occasion (that is, more than 90% of the people played on different dates). Furthermore, every player in the top scores list played over 800 games (that's over 53 hours without including the time they spent waiting for a partner!). This undoubtedly attests to how enjoyable the game is.

User Comments
To give a further sense for how much the players enjoyed the game, we include below some quotes taken from comments submitted by players using a link on the website:

> "The game itself is extremely addictive, as there is an element of pressure involved in beating the clock, a drive to score more points, the feeling that you could always do better next time, and a curiosity about what is going to come up next. I would say that it gives the same gut feeling as combining gambling with charades while riding on a roller coaster. The good points are that you increase and stimulate your intelligence, you don't lose all your money and you don't fall off the ride. The bad point is that you look at your watch and eight hours have just disappeared!"
>
> "One unfortunate side effect of playing so much in such a short time was a mild case of carpal tunnel syndrome in my right hand and forearm, but that dissipated quickly."
>
> "This game is like crack. I've been Peekaboom-free for 32 hours. Unlike other games, Peekaboom is cooperative (rather than competitive)."

EVALUATION: ACCURACY OF COLLECTED DATA
The usefulness of Peekaboom as a data-collection method rests in the quality of the data we collect. Although the design of the game inherently ensures correctness of the data, we wanted to test whether it is as good as what would be collected directly from volunteers in a non-game setting. To do so we conducted two experiments to test first the accuracy of the bounding boxes we defined, and second the utility of the pointing behavior in the game.

Notice that these experiments are meant to analyze the correctness of the data, and not whether such data can be used to train computer vision algorithms. The usefulness of data about location of objects for training computer vision algorithms has been previously established [13].

Experiment 1: Accuracy of Bounding Boxes
In the first experiment, we tested whether the bounding boxes for objects within an image that are calculated from Peekaboom are as good as bounding boxes people would make around an object in a non-game setting. We selected at random 50 image-word pairs from the data pool that had been successfully played on by at least two independent pairs of people. The images selected all had nouns as their word (as opposed to text in the image, or an adjective, etc.; see Figure 3). All the images chosen had the word refer to a single object in the image. For each image, Peekaboom data was used to calculate object bounding boxes using the method explained in previous sections.

We then had four volunteers make bounding boxes around the objects for each image, providing us with 200 bounding boxes drawn by volunteers. The volunteers were asked, for each image, to draw a bounding box around the object that the word referred to. We then selected at random one of the four volunteer's bounding boxes for each image, so as to end up with one volunteer-generated bounding box for every one of the 50 images.

Finally, we tested the amount of overlap between the bounding boxes generated by Peekaboom and those generated by our volunteers. The amount of overlap was determined using the formula:

$$\text{OVERLAP}(A,B) = \text{AREA}(A\cap B) / \text{AREA}(A\cup B),$$

where A and B are the bounding boxes. Notice that if A=B then OVERLAP(A,B)=1 and if A is disjoint from B then OVERLAP(A,B)=0. We calculated the average overlap across the 50 images, as well as the standard deviation.

Results
On average, the overlap between the Peekaboom bounding boxes and the volunteer-generated ones was 0.754, with standard deviation 0.109. This means that the Peekaboom bounding boxes were very close to those generated by the volunteers. To illustrate, we show in Figure 9 the bounding box that obtained the lowest overlap score (0.552).

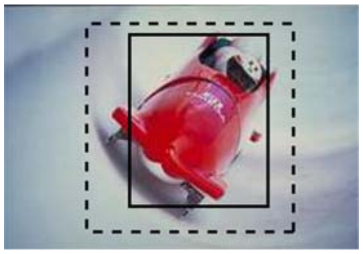


Figure 9. Experiment image with lowest overlap between a volunteer generated bounding-box (solid lines) and one generated by Peekaboom (dashed lines).

Given that Peekaboom was not directly built to calculate bounding boxes, this shows the wide applicability of the data collected.

Experiment 2: Accuracy of Pings

In the second experiment, we tested whether the object pointers that are calculated from Peekaboom are indeed inside the objects. As in the previous experiment, we selected at random 50 image-label pairs from the data pool that have been successfully played on by at least two independent pairs of people. The images selected all had the word as a "noun" (as opposed to as text in the image, or an adjective, etc.; see Figure 3). All the images chosen had the word refer to a single object in the image. For each image, Peekaboom data was used to calculate object pointers using the method explained in previous sections.

We then asked three volunteer raters to determine, for each pointer, whether it was inside the object or not. The raters were shown examples of pointers inside and outside the object and were told that "near an object" does not count as inside the object.

Results

According to all the raters, 100% of the pointers were inside the object referred to by the word. This gives positive evidence that ping data is accurate, especially since it was calculated using such a simplistic algorithm.

GENERALIZING OUR APPROACH

The approach presented in this paper, solving a problem by having people play games online, can be generalized to many other problems in Artificial Intelligence. In follow-up work, for example, we have created two other games, Verbosity [3] and Phetch [2], in which players solve problems that computers cannot yet solve. Verbosity collects common-sense facts to train reasoning algorithms. For instance, for the word "milk," the game outputs facts such as "it is white," "people usually eat cereal with it," etc. Verbosity is a two-player game in which one player attempts to make the other say a target word (e.g., "milk") without using the word. They do so by saying many facts without using the word itself in their statements (e.g. "it is a white liquid."). The underlying game mechanism of Verbosity is similar in nature to that of Peekaboom.

Much like designing an algorithm to solve a problem, designing a game to harness valuable human cycles is to a large extent an "art": problems usually require a specifically tailored game. In addition to an original idea, creating such a game also depends on a broader set of criteria including looks (the fluidity of the game graphics), ease of use (an intuitive user interface), cognitive load (the amount of user attention required to play the game), and action (the extent to which the game absorbs the user in the experience). All of these aspects have been treated in this paper and we believe many of the techniques here presented generalize to creating other games with a purpose. Finally, we believe that these design principles, like the scientific method, don't just provide ideas, but a *way* of thinking: games provide a valuable vehicle to solve problems that computers cannot yet solve.

ETHICAL CONSIDERATIONS

As with all systems soliciting input from humans, we must address the ethical issues behind the usage of the collected data. Towards this end, we inform the players of the game's purpose on the Peekaboom website — players participate willingly and knowingly. Indeed, many people play because they like the fact that the game has a purpose.

Furthermore, we state on the record that the game's purpose is to obtain accurate segmentations of objects from backgrounds and to train computer vision algorithms to recognize simple objects. We have no intention of applying our data towards, for example, military surveillance.

RELATED WORK

We have presented a method for annotating arbitrary images and we have presented evidence that it produces high-quality data. We now survey the related work.

The ESP Game

As mentioned before, the ESP Game [1] is two-player game that collects word labels for arbitrary images. Peekaboom is similar to the ESP Game and in fact was inspired by it. We consider Peekaboom an extension of ESP. Whereas ESP gives data to determine which objects are in the image, Peekaboom can augment this data with information about *where* in the image objects are located.

In terms of game mechanics, Peekaboom is different from the ESP Game in several ways. First, Peekaboom is *asymmetric*: whereas both players in the ESP Game are performing the same role, players of Peekaboom alternate in performing different roles. Second, Peekaboom allows a significantly higher level of interaction among the players. Whereas in the ESP Game, players cannot communicate at all, in Peekaboom one of the players can freely communicate with the other. Third, the usage of hint buttons has proven very successful in Peekaboom, and such buttons could as well be incorporated into ESP.

Such differences in game mechanics reflect the difference in purpose of Peekaboom and ESP.

The Open Mind Initiative

Perhaps less so, Peekaboom is also similar (at least in spirit) to the Open Mind Initiative (e.g., [11,12]), a worldwide effort to develop "intelligent" software. Open Mind collects data from regular Internet users (referred to as "netizens") and feeds it to machine learning algorithms. Volunteers participate by answering questions and teaching concepts to computer programs. Peekaboom is similar to Open Mind in that we use regular people on the Internet to annotate images. However, as with the ESP Game, we put much greater emphasis on our method being fun.

We don't expect volunteers to annotate millions of images on the Web: we expect images to be annotated because people want to play our game. Whereas a typical Open Mind activity would ask participants to point to the object in question, we *transform* the activity into a two-player game in which players are not even asked to point to the object; they do so only as a side effect of playing the game.

LabelMe

LabelMe [9] is a web-based tool for image annotation. Anybody can annotate data using this tool and thus contribute to constructing a large database of annotated objects. The incentive to annotate data is the data itself. You can only have access to the database once you have annotated a certain number of images. The main difference between Peekaboom and LabelMe is the game aspect. Whereas LabelMe simply asks users to annotate an image, Peekaboom transforms the process into an enjoyable game. LabelMe relies on people's desire to help and thus assumes that the entered data is correct. On the other hand, Peekaboom has multiple mechanisms to prevent players from polluting the data.

Interactive Machine Learning

Another area of related work is that of interactively training machine learning algorithms (e.g., [6]). In these systems, a user is given immediate feedback about how well an algorithm is learning from the examples provided by them. As with LabelMe, Peekaboom differs from these systems in the gaming aspect as well as in the assumption that our users are interested in training an algorithm.

CONCLUSIONS AND FUTURE WORK

Peekaboom is a novel, complete game architecture for collecting image metadata. Segmenting objects in images is a unique challenge, and we have tailored a game specifically to this end. In the very near future, we would like to make our 1,000,000+ pieces of data available to the world by formatting it as an image segmentation library.

Like the ESP Game, Peekaboom encompasses much more than just a Java applet delivered from a website. Rather, the ideas behind the design and implementation of the game generalize to a way of harnessing and directing the power of the most intricate computing device in the world — the human mind.

Some day computers will be able to segment objects in images unassisted, but that day is not today. Today we have engines like Peekaboom that use the wisdom of humans to help naïve computers get to that point. The actual process of making computers smarter given segmentation metadata is beyond the scope of this paper, since it would require a far more sophisticated interpretation of the data than the simple bounding box derivation we have presented. Thus, we see great potential in future work at the crossroads of human-computer interaction and artificial intelligence, where the output of our interactive system helps advance the state of the art in computer vision.

ACKNOWLEDGEMENTS

We thank Aditya Akella, Laura Dabbish, Takeo Kanade, Ryan Staake, Manuela Veloso, and the anonymous CHI 2006 reviewers. This work was partially supported by the National Science Foundation (NSF) grants CCR-0122581 and CCR-0085982 (ALADDIN), and by a generous gift from Google, Inc. Luis von Ahn was partially supported by a Microsoft Research Graduate Fellowship.

REFERENCES

1. von Ahn, L., and Dabbish, L. Labeling Images with a Computer Game. In *ACM Conference on Human Factors in Computing Systems (CHI),* 2004, pages 319-326.
2. von Ahn, L., Ginosar, S., Kedia, M., Ruoran, L. and Blum, M. Improving Accessibility of the Web with a Computer Game. To appear in *ACM Conference on Human Factors in Computing Systems (CHI Notes),* 2006.
3. von Ahn, L., Kedia, M. and Blum, M. Verbosity: A Game for Collecting Common-Sense Facts. To appear in *ACM Conference on Human Factors in Computing Systems (CHI Notes)*, 2006.
4. Barnard, K., and Forsyth, D. A. Learning the Semantics of Words and Pictures. *International Conference of Computer Vision*, 2001, pages 408-415.
5. Duygulu, P., Barnard, K., de Freitas, N., and Forsyth, D. A. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. *European Conference on Computer Vision,* 2002, pages 97-112.
6. Fails, J. A., and Olsen, D. R. A Design Tool for Camera-Based Interaction. In *ACM Conference on Human Factors in Computing Systems (CHI),* 2003, pages 449-456.
7. Fleck, M. M., Forsyth, D. A., and Bregler, C. Finding Naked People. *European Conference on Computer Vision*, 1996, pages 593-602.
8. Flickr Photo Sharing Service. http://www.flickr.com.
9. Russell, B.C., Torralba, A. Murphy, K.P. and Freeman, W.T. LabelMe: a database and web-based tool for image annotation. *MIT AI Lab Memo* AIM-2005-025, September, 2005.
10. Scheniderman, H. and Kanade, T. Object Detection Using the Statistics of Parts. *International Journal of Computer Vision,* 2002.
11. Stork, D. G. and Lam C. P. Open Mind Animals: Ensuring the quality of data openly contributed over the World Wide Web. *AAAI Workshop on Learning with Imbalanced Data Sets,* 2000, pages 4-9.
12. Stork, D. G. The Open Mind Initiative. *IEEE Intelligent Systems and Their Applications, 14-3,* 1999, pp. 19-20.
13. Torralba, A., Murphy, K. P. and Freeman, W. T. The MIT CSAIL Database of objects and scenes. http://web.mit.edu/torralba/www/database.html
14. Torralba, A. An example of the importance of context in vision. http://web.mit.edu/torralba