SOILIE: A Computational Model of 2D Visual Imagination

Vincent Breault (breault_vincent@gmail.com)
Sébastien Ouellet (sebouel@gmail.com)
Sterling Somers (sterling@sterlingsomers.com)
Jim Davies (jim@jimdavies.org)

Institute of Cognitive Science, Carleton University
1125 Colonel By Drive, Ottawa, Ontario K1S 5B6 Canada

Abstract

The Science of Imagination Laboratory Imagination Engine (SOILIE) is a program designed to imagine 2D scenes the same way people do. It is composed of three modules: The Oracle of Objects, Visuo, and a renderer. In the process of creating a novel image, SOILIE creates image content, places objects within said image and renders the final product. Although work remains to be done, the engine is capable of rendering images with coherent content and placement.

Keywords: imagination; modeling; procedural generation; cognitive science; artificial intelligence; graphics; visual reasoning

Introduction

The term "imagination" is typically used to describe two different kinds of human cognitive processes: first, the ability to be creative in general, and second, the ability to generate simulations of world states, either real or fabricated, in the mind. This paper focuses on the latter, in particular visual imagination.

Imagination is implicated in a great number of cognitive processes. People will often imagine scenes when hearing a story or reading a novel, when planning physical action, when recalling previous experiences, fantasizing about the future, and when dreaming (Davies, Atance, Martin Ordas, 2011). Although visual imagination is often thought to be identical with visual mental imagery, we view the rendering of a mental image as a final, optional stage. The process of rendering an imagined scene into neural "pixels" (colors at particular locations) must be preceded by (mostly unconscious) decisions regarding what is to be placed in the image, and where. For example, if one is asked to picture "a cat under a car," one is likely to also picture a road or driveway beneath the car and the cat. How does an intelligent agent know to put a road beneath a car, or a sky above a mountain range?

In this paper we describe a cognitive model of visual imagination, and its implementation as a Python computer program called SOILIE (Science of Imagination Laboratory Imagination Engine).

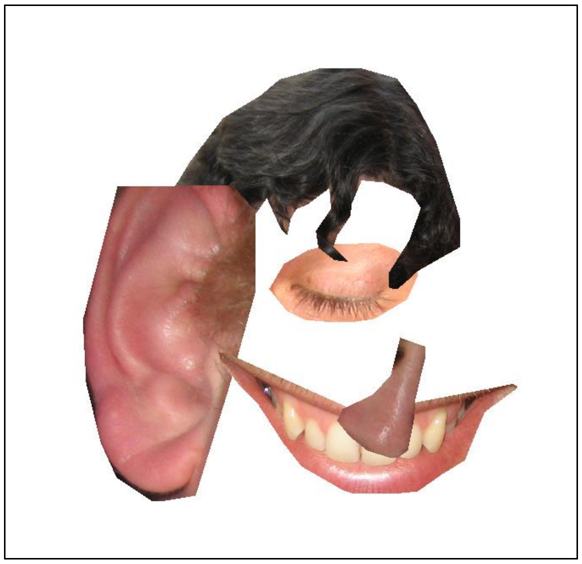

Figure 1: SOILIE's imagined output given the query 'eye'

Theory of Visual Imagination

We are starting by modeling a relatively simple task: taking as input a single word (the "query") and creating a static 2D image that contains other elements that are likely to appear with the object described by the query word. The imagined scene that is output does not contain any motion, physical simulation, or depth. It does not change over time, nor does it attempt to be creative in the sense of generating a surprising or particularly interesting result. The goal is to create realistic yet novel images.

In our model, the agent takes a single word as the impetus to imagine, e.g., "car." The task of the agent is to imagine a car in a realistic scene. Using visual long term memories, the agent populates the scene with other elements that are likely to appear in an image with a car, such as a road. Once the agent has decided what else should appear in the image and where those elements should be placed, then the entire scene is rendered into pixels, that is, points of colored light at certain locations. In the case of people, different neurons in the retino-topic visual cortex increase in firing frequency, forming a mental image on which perceptual processes can then operate (Kosslyn, 1994; 1995) (we acknowledge that the very existence of mental imagery in human beings remains controversial: see Pylyshyn (1994) for a counterargument). In our computer implementation, this means creating a collage on the screen of objects generated from pixels taken from various images in a database.

We assume that the sub-image of the query object appears in the center of imagination's "visual field," and the other elements are placed around it appropriately. For example, if imagining a car, the car would appear in the center of the mind's eye, the road would go beneath it, the sky would appear above it, and a person might be beside it.

System: SOILIE

The Science of Imagination Laboratory Imagination Engine (SOILIE) is an implemented Python program composed of multiple modules that together create a 2D visual scene from a user-input query. In its current state, the engine takes a single word query as input and returns an imagined 2D image containing several elements related to the initial query. The ultimate goal of SOILIE is to create imagined visual scenes in the same way that humans do, and to produce imaginings that resemble human imaginings given the same input.

SOILIE uses large databases of labeled images as its proxy for human visual experience. For each image, labels are associated with particular pixels so that we know where labels appear in the images. For example, if the label is "car," then the database knows the rough outline of the car in the image, and we assume that all the pixels within that outline represent light coming from the car to the camera.

The various modules of SOILIE use these datasets of large quantities of labeled images to extract regularities such as what type of labels tend to be found together and the spatial relationships between labels. These extracted spatial relationships are SOILIE's models of spatial memory. The three subsystems will be presented: The Oracle of Objects, Visuo, and the Renderer.

The Oracle of Objects

The first module is the Oracle of Objects (Astudillo, 2009). Using data gathered from the game Peekaboom, this module has information on image content. Peekaboom's data is a collection of over fifty thousand labeled images with over ten thousand labels that are used as representation of visual memory. This database is the combination of two online games: the ESP Game and Peekaboom (Von Ahn, Liu, & Blum, 2006). These games are interactive systems where players are paired over a server and each presented with the same image. Without being allowed to communicate, the players try to enter the same words to describe objects in the images. Their answers are compared. When they enter the same word, the label is kept and stored with the image in the database.

This information was used in a second game called Peekaboom, which used a similar strategy to find out where in the images the objects were. One player would use mouse clicks to reveal parts of an image to another player, whose goal was to guess the label given to the first player. When the second player guesses the right word, we assume that the parts of the image revealed represent where the object is in the image.

The result is that labels are associated with an image and as point clouds on specific locations on the picture, representing the location of the object. These games provide the advantage of gathering accurate and diverse labeling data on very large set of images in a fast and efficient way (von Ahn & Dabbish, 2004).

The Oracle of Objects uses a matrix of co-occurrence relations derived from Peekaboom's image content information. This matrix holds the frequencies at which two labels will appear in the same picture in the database. Taking the top *n* labels that co-occur with a particular query allows the module to determine which other objects should be included in the imagined image. This method is called the "top-*n* model" (see Vertolli & Davies, under review, for our attempts to improve on this model).

In SOILIE, the Oracle takes the single word query, such as the word "sky" and uses the database's co-occurrence matrix to find the *n* items that co-occur with it most often throughout all the labeled images. The specific number of objects returned by the Oracle is ten but SOILIE takes only the first three or four, as the likelihood of producing incoherent images increases with the number of objects. Furthermore, many objects would clutter the picture. This parameter can easily be changed in the program. This module's output thus determines descriptive content of the image that is to be created. For example, given the query "sky" and a target number of objects of four, the Oracle would return the following: ['water', 'building', 'cloud', 'mountain']. The next module, Visuo, takes the objects from this output to determine their relative positioning.

Visuo

SOILIE uses the Visuo module (Davies & Gagné, 2010; Somers, Gagné, Astudillo & Davies, 2011) to find the appropriate placement for each of the labels in the picture. Visuo is a cognitive model implementing a theory of quantitative spatial memory and the learning involved with that memory, generation of imagined spatial magnitudes, and analogy. For SOILIE, Visuo finds the appropriate distance and angle between the query and each of the labels chosen by the Oracle. Visuo takes in two labels, query and object, and returns a location in the 2D canvas of the object in relation to the query at the center. The distance is represented as a percentage of space from center of the image and the border of the frame.

The Peekaboom database contains information from which we can easily infer distance and angle between any two labels that co-occur in an image. Visuo was trained on these for every pair of labeled objects, e.g., man – hat. The input structures contain a pair of labels, linked by the term "above." We acknowledge that not everything is above everything else; we use it only as a generic spatial relation word. Each example of a pair of labels also contained values for the attributes of distances and angle gathered from each instance from the game. This is specifically represented in the training file as follows:

```
name = [hat] above man
lexical_category = "preposition"
distance = 0.4253361716
angle = -93.909154542
```

There would be one such data structure for every instance of a hat being above a man found in the database of images. The angles and distances are not stored as crisp numbers, but rather as a vector of fuzzy membership values for particular ranges. So a single angle would be stored as a vector of fuzzy membership values (between 0 and 1) within a set of fuzzy number categories of angles in this set: [-157.5, -135, -112.5, -90, -67.5, -45, -22.5, 0, 22.5, 45, 67.5, 90, 112.5, 135, 157.5, 180]. So, for example, 160 degrees is more a member of the 157.5 fuzzy set, and less a member of the 135 fuzzy set, etc.

Similarly, distance is stored as memberships in the following categories of distance, which approximates the logarithmic scale [0, 2, 5, 10, 20, 35, 60, 100, 160, 250, 400, 600, 900, 1350, 1800]. These numbers represent arbitrary mental units and not any specific objective metrics. This models the fact that people do not need any specific culturally invented unit of measure to represent visuospatial magnitudes (Gagné & Davies, 2010). When queried, these fuzzy distributions are de-fuzzified and return a crisp number indicating the value for the angle in terms of relative degree with the main query and distance as a percentage from the maximum distance to border of the image frame.

Each instance from the example files is transformed into a data structure in Visuo called an exemplar, and some prototype memory is modified (or created if it does not yet exist). These prototypes contain a fuzzy distribution of each of the examples' attribute values for both the distance and angle. For instance, the prototype "[hat] above man" contains the distribution of values for each "[hat] above man" example. The prototypes can thus be thought of as an average of all experiences of a given pair of labels.

With the prototypes created for each pair of labels in the Peekaboom database, Visuo has a large trained set of distances and angles for pairs of labels that can be used to determine placement of elements in SOILIE's imagined image. In order to determine the value for specific pair of labels, Visuo looks in its database for the specific prototype matching the query. Once it has found it, it will transform the fuzzy distributions of angle and distance into crisp quantitative numbers, which SOILIE uses to place elements in the image in relation to the query, which is in the center. Following the previous example of the query "sky" and the returned elements ['water', 'building', 'cloud', 'mountain'], SOILIE would query Visuo with the following input:

```
[sky] above water
[sky] above building
[sky] above cloud
[sky] above mountain
```

This would, in turn, return the respective distance and angle for these pair of objects. This output takes the following form, the first number being the angle in degrees:

```
#sky
sky-water -86.6024480504 0.938798453022
sky-building 71.0785158959 0.879984377514
sky-cloud 83.6170496771 0.371586153984
sky-mountain 97.3158893841 0.55739012773
```

After this has been done for the each included element, SOILIE knows what elements will be in the image, and where they should appear. This scene description is sent to the renderer to generate an image that can be seen on a computer screen.

Lastly, Visuo also has the ability to estimate angles and distance of pairs of label on which it has not yet received training. When it is asked to get special information on a novel pair of labels, Visuo will use the WordNet database (Fellbaum, 1998) to determine the meaning of the unrecognized words. It uses WordNet's semantic similarity information to find a *known* pair that correlates highest with the query. For example, if it does not have a prototype for "Chevrolet above pug" it might return "car above dog," for which it does have a prototype.

Once the most correlated pair has been found, Visuo will use its attribute distribution in place of the query's lack of data. This allows the program to return approximate information for labels it does not know based on semantic comparison with what it does know.

Renderer

The last module is tasked with displaying the imagined image. It takes as input the elements and their locations on the canvas from two previous modules. The main query is then placed in the center of the image and each of the other items is placed around it according to the angle and distance specified.

To get the necessary pixels for the rendering of the picture, this module uses the images from the LabelMe database. LabelMe is a "web-based annotation tool that allows easy image annotation". (Russell, Torralba, Murphy, & Freeman, 2008). This provided an effective tool to collect quantitative information on annotated objects in the database. LabelMe provides a drawing interface for users to specify the boundaries of objects in an image as well as a label. LabelMe's bounding boxes provide better outlines of objects than the point clouds in the Peekaboom database. LabelMe also has the advantage of using photographs from the Web as opposed to all web images, which include cartoons and logos. This allows SOILIE to use real-world pictures of objects when rendering output.

The LabelMe labels are indexed in a preprocessing stage, before SOILIE is first run, where each label is associated with images containing the label. An image is chosen at random from the list corresponding to the queried label, and the outline information is extracted. The corresponding image file is then loaded into a PyGame 1.9 instance. In order to avoid overly large objects taking too much of the visual space, image files of more than 500 pixels wide are halved in size. This prevents objects such as skies from hiding other objects in the scene or unusually big items such as ears from appearing too much out of proportion. A polygon is drawn from the vertices corresponding to the LabelMe outline, and the positions of the pixels inside the polygon are kept as a mask, which is then applied to the LabelMe image. The pixels under the mask are finally displayed in the rendering window. The location is determined by a vector given by Visuo, the initial point of the vector is mapped to the center of the picture, and the center of the bounding box of the pixels is mapped to the terminal point of the vector.

Evaluation

The next section will qualitatively examine the strengths and weaknesses of the current model through examples from its output.

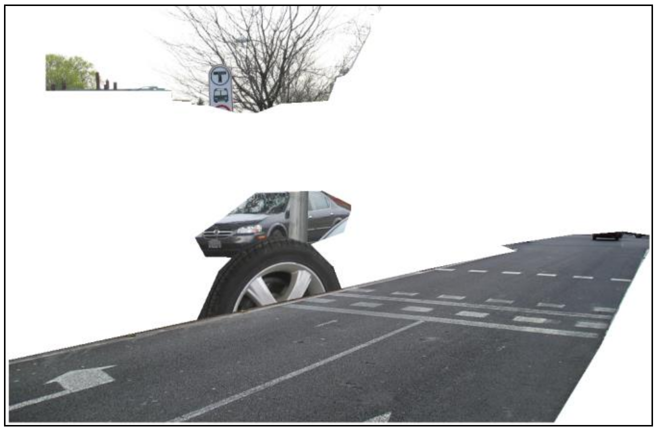

Figure 2: Result of the query ‘car’

SOILIE is capable of generating a wide variety of images. The Oracle of Objects’ use of the coherence matrix as well as Visuo's training on a large database and its ability to approximate an appropriate value for distance and angle through analogy allows SOILIE generates somewhat coherent images. This has been supported with quantitative evaluation in Somers, Gagne, Astudillo, and Davies (2011).

Every Figure in this paper depicts images generated by SOILIE. Figure 2 shows the resulting rendered image based on the query “car”. Given such a query, SOILIE returned the following labels [‘sky’, ‘wheel’, ‘road’]. This is so because they are labels that are often seen in the same image as a car and thus were highest in the co-occurrence matrix. From there, Visuo assumed the car to be in the center and returned coherent placement for the other labels. The image demonstrates the system’s ability for coherent placement, as the sky was placed above the car and both the road and the wheel under it, like any human might expect from such a scene.

Furthermore, the Oracle of Objects returns sets of objects with a relatively high coherence. Table 1 shows several examples of the Oracle’s output when given certain queries. As one can see, the use of the top-*n* items in the co-occurrence matrix permits the creation several coherent of sets of item.

Table 1: Sample label output.

Query	Returned Labels
Sky	[‘water’, ‘building’, ‘cloud’, ‘mountain’]
Tree	[‘building’, ‘water’, ‘house’, ‘grass’]
Table	[‘woman’, ‘people’, ‘food’, ‘chair’]
Dog	[‘grass’, ‘ear’, ‘puppy’, ‘man’]
Bed	['window', 'woman', 'sleep', 'room']
Mouse	[‘rat’, ‘computer’, ‘animal’, ‘keyboard’]

It should however also be noted that in its current state, SOILIE has very low context sensitivity when the query has strongly correlated words from different contexts, such as homonyms. One such example is the word “mouse,” which can refer to both computer mouse and the animal. The engine currently has no way of differentiating these two contexts and if two labels are correlated enough, they will come out in the image (we address this problem in our lab’s forthcoming paper: Vertolli & Davies, under review). Figure 3, in which a computer mouse is partially hidden by a groundhog and a deer, clearly demonstrates the integration of the two contexts of mouse in a single image. Queried with the word mouse, the Oracle returns ['rat', 'computer', 'animal', 'keyboard'], showing no discrimination of the mouse as animal and as computer hardware. This is problematic as it can create serious incoherence in resulting images.

Furthermore, it has no concept of meronyms. This means that the system does not know that a bird has a beak or that faces have noses. But because they are strongly correlated in the database, it is highly probable to get an image containing both a picture of a bird and one of a beak. Figure 2 is an example of this. The query ‘car’ has returned wheel but one can clearly see that the car pixels returned already feature wheels, so the additional wheel was unnecessary. Table 1 also demonstrates this problem as the query dog returned the label ‘ear’ even though it is not a necessary addition as ears are already parts of dogs. Our laboratory’s future work will address this problem as well. Also note that `puppy’ was returned, showing that SOILIE also fails to ignore synonyms.

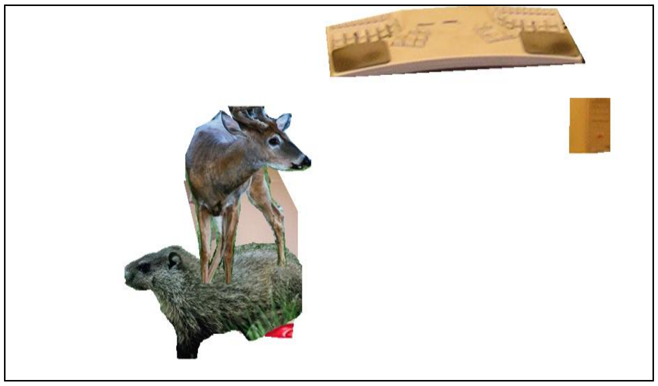

Figure 3: Result of the query 'mouse.' Note that it appears that an image of a groundhog was incorrectly labeled as 'rat' in the LabelMe database.

Another drawback is Visuo's current lack of information on diverse attributes. Although it currently has a lot of data on distances and angles, it knows nothing of other attributes such as size or depth. For instance, it can tell that a cap is close to a head but in the actual rendering of the scene, there is no data on the relative size of the cap and the head, possibly creating images with set of objects with poor proportions. Figures 1, 2 and 3 all demonstrate SOILIE's lacking of the size attribute. In figure 2, the wheel is significantly bigger than it should be for the size of the car and Figure 3 shows a computer that is disproportionately too small for the size of the keyboard and the mouse. Using the current database for Visuo training, Peekaboom, such information would be hard to acquire as it does not contain boundary information about its labels or depth information, rendering the extraction of relative size quite difficult without sophisticated computer vision techniques.

Figure 2 also demonstrates some of the rendering errors. Because the rendering engine uses the center of the object to determine placement, it ignores the object's boundaries. In Figure 2 the road cuts the wheel in half and in Figure 1 by the nose overlapping with the mouth. This makes for images with inappropriately overlapping objects.

Another problem that arises in the rendering stage is that when the Oracle of Objects returns the label 'ear' when queries with `dog'. The renderer finds an image of an ear and places it on the canvas. Unfortunately, the current system has no way to determine whether the picture file it is using contains a dog ear or a human ear or, again as in Figure 2, if the mouse file contains an image of an animal or a computer hardware.

Figure 4 demonstrates quite eloquently most of the current rendering module's shortcomings. Although the 'ear' was placed above and to the side of the dog's head, it overlaps greatly with it because the object's boundaries are not taken into consideration. It can clearly be seen that the engine used a human's ear for a dog's ear and the lack of objective size information made the ear significantly bigger than the dog itself.

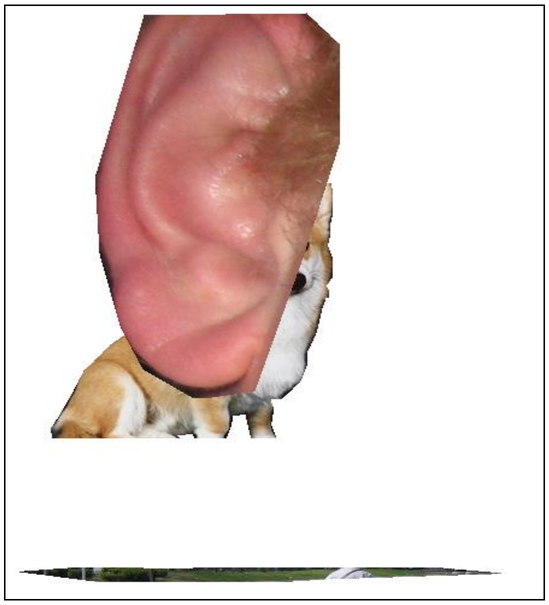

Figure 4: Result of the query 'dog'

Discussion

The first implemented model of mental imagery was done by Kosslyn and Shwartz (1977). The system used a polar coordinate pixel system that successfully modeled some psychological findings of mental imagery, including 1) taking longer to generate more complex images, boundaries of the mind's eye, rotation, scaling, and incremental increase in detail according to attention shifting. The system could work with two images: a chair and a car. This model can be included in a range of implemented "array theories" that eventuate in pixel placement, some explicitly symbolic (e.g., Glasgow & Papadias) and some connectionist (e.g., Stucki & Pollack, 1992). These models do not speak to how elements are selected for inclusion in the scene, but provides a more realistic model of the final stage of imagination. In contrast, our approach uses the final image that is output primarily for demonstration purposes. Future work might incorporate a more realistic model of the visual cortex for the rendering stage.

Cognitive scientists tend to agree that humans do not store bitmap-like (depictive) representations in long-term memory, but rather proposition-like (descriptive) encodings of scenes (Kosslyn, 1994). Our system uses photos from LabelMe to get elements for display in the imagined scene. We do not believe that human beings do it this way. However, how people generate depictive representations from descriptive ones is a mystery.

Computer scientists have created programs that design and render (Ebert et al., 2003). These systems are not designed to be cognitive models. In this field it is typically not called "imagination", but goes by names such as procedural generation, synthesis, dynamic generation, procedural modeling, procedural synthesis, and visualization.

In terms of content, such systems tend to focus on one of the following categories: faces and people, plants, terrain, planets, interiors, or cities. What is more interesting about these systems are the methods used: *either grammars*, explicit knowledge, or fractals.

Grammars use explicitly coded rules that describe acceptable expressions to determine what gets placed where. For example, chairs should appear surrounding a table.

Explicit knowledge might use templates or cases to define what things should look like. For example, there might be a rule that says that all rooms must have doors. As in many AI applications, explicit knowledge of this kind tends to be brittle, expensive to encode, and generates contradictions as the knowledge base grows.

Fractal methods, which are primarily used for natural systems such as plants or terrains, use mathematical descriptions. Fractals are self-similar at different scales. The branching of a tree from a trunk resembles the branching of smaller branches from the original branch, and so on. Many natural phenomena, particularly ones that maximize area in a finite space, can be modeled with fractals.

Conclusion

SOILIE is able to take a user input of a single word and generate a rendered image of a 2D scene depicting the query word and several associated images, in more or less the correct places. Although the placement of sub-images in the final canvas makes sense if you know what it is doing, the images are admittedly unlike our internal experiences of imagery, and appear rather like surrealistic art pieces. Future work should improve the images.

In particular, the problems faced are 1) incoherent label choice (particularly with homonyms), 2) the system's misunderstanding of part-whole relationships (meronyms), 3) a misunderstanding of kind with respect to parts and wholes (a dog's ear versus a human ear) and 4) lack of information about size. Future work will address these problems, as well as improve the nature of the output using pixel interpolation to fill in blank spaces (Hays & Efros, 2007; Feiner, 1985), to create a more realistic neural model of the rendering.

References

Astudillo, C. (2009). *Co-Occurrence of Objects in an Image*. Website. Retrieved from http://ing.utalca.cl/~castudillo/research/pkb/co_ocurrence/form.php

Davies, J., Atance, C. & Martin Ordas, G. (2011). A framework and open questions on imagination in adults and children. *Imagination, Cognition, and Personality, Special issue on mental imagery in children*. 31:1-2, 143-157.

Davies, J. & Gagné, J. (2010). Estimating quantitative magnitudes using semantic similarity. *Conference of the American Association for Artificial Intelligence workshop on Visual Representations and Reasoning* (AAAI-10-VRR), 14--19.

Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., & Worley, S. (2003). *Texturing and modeling, 3rd Edition: A procedural approach*. San Francisco, CA: Elsevier Science.

Feiner, S., (1985). APEX: An Experiment in the Automated Creation of Pictorial Explanations. *IEEE Computer Graphics and Applications, 5(11)*, 29-37

Fellbaum, C. (Ed.) (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Glasgow, J., & Papadias, D. (1992). Computational imagery. *Cognitive Science, 16*, 355-394.

Hays, J., Efros, A. A. (2007). Scene Completion Using Millions of Photographs. *ACM Transactions on Graphics (SIGGRAPH 2007), vol. 26*, no. 3

Kosslyn, S. M. (1994) *Image and Brain: The Resolution of the Imagery Debate*. MIT Press, Cambridge, MA.

Kosslyn, S. M., Thompson, W. L., Kim, I. J., & Alpert, N. M. (1995). Topographical representations of mental images in primary visual cortex. *Nature, 378*(6556), 496-498.

Kosslyn, S. M. & Shwartz, S. P. (1977). A simulation of visual imagery. *Cognitive Science* 1, 265--295.

Pylyshyn, Z. W. (1994). Mental pictures on the brain. *Nature*, 372, 289-290.

Russell, B. C., Torralba, A., Murphy, K., Freeman, W., (2008), LabelMe: a database and web-based tool for image annotation, *International Journal of Computer Vision, vol. 77,* Issue 1-3.

Somers, S., Gagné, J., Astudillo, C., & Davies, J. (2011). Using semantic similarity to predict angle and distance of objects in images. *In Proceedings of the 8th ACM Conference on Creativity & Cognition (pp. 217-222)*. Atlanta, Georgia.

Stucki, D. J., & Pollack, J. B. (1992). Fractal (reconstructive analogue) memory. *Proceedings, fourteenth annual conference of the cognitive science society*. Hillsdale, NJ: Erlbaum

Chen, T., Cheng, M., Tan, P., Shamir, A., Hu, S., (2009) Sketch2Photo: internet image montage, *ACM Transactions on Graphics (TOG)*, v.28 n.5,

Vertolli, M. O. & Davies, J. (under review) Visual imagination in context: Retrieving a coherent set of labels with Coherencer. Submitted to ICCM 2013.

von Ahn, L., and Dabbish, L. (2004) Labeling Images with a Computer Game. *ACM Conference on Human Factors in Computing Systems (CHI)*

von Ahn, L., Lui. R., & Blum. M. (2006) Peekaboom: A game for locating objects in images. *In Proceedings of the SIGCHI conference on Human Factors in computing system (pp. 55-64).* AC