vr-rl documentation (https://github.com/kjemery/vrrl)

The goal of this project was to build a computational account of how scene semantics are leveraged in visual search, specifically when an agent searches for a given object in a naturalistic scene. The following list of functions model visual search by predicting a sequence of fixations based on object cooccurrence statistics from a database of natural images (ADE20K). The object cooccurrences serve as the energy function to a probabilistic graphical model (pgm).

To demonstrate a proof of concept, the following algorithm currently works to predict a series of fixations as a series of fixated ROI's in an image. Therefore, each image fed to the algorithm is first divided into a set of ROI's or a grid of "patches" of a set size. A separate, local pgm operates within each patch to maintain an account of which variables (objects) are present (1) and absent (0) in that region of space. Subsequent fixations are chosen based on which patch has the highest marginal probability for a given target object based on the objects with evidence of being present in that patch. Evidence for the objects within the patch at the center of fixation is equivalent to complete certainty of their states, and the amount of evidence falls off exponentially with a patch's distance from the center of fixation. During search, evidence gained does not decay, meaning that the highest likelihood or evidence for a given object is maintained during the duration of a search episode. Furthermore, the same patch will not be fixated twice (i.e., inhibition of return), for if the target object is not detected in that patch upon first fixation, this indicates that it is not present in that patch. Fixations continue until the target is located or the maximum number of fixations is reached (e.g. 20).

pseudocode for algorithm (fixation_prediction_final.ipynb):

Step 1: define variables

n_states = number of states per object (always two for 0 and 1 in the present case)
pixels_in_patch = patch size in pixels

Step 2: load scene data

data_scene = load json
image = load image

Step 3: calculate cooccurrence mat

get_cooccurrence_mat(database, data_scene)

Step 4: create graph

create_graph(object_classes, cooccurrence_mat)

Step 5: create scene data frames for pgm

get_centroids(data_scene, object_labels)
create_scene(image, pixels_in_patch, centroid, object_labels)

Step 6: predict fixation sequence

get_fixation_seq(latent_scene, observed_scene, graph, target, sigma_likelihood)

If last_fixation[target] == 1
  result = 'correct'

Step 7: visualize fixation sequence

visualize_search(image, pixels_in_patch, fixation_seq, target, filename)

fixation_prediction_final.ipynb functions:

get_cooccurrence_mat:

Calculates object cooccurrences from a database of natural images (ADE20K). These calculations are made by counting each time a pair of objects appears in the same image. These counts create a cooccurrence matrix that represents the number of times each possible pair of objects in the database appears together across all images, and consequently the probability for all possible configurations of present and absent for the pair (i.e. (0,0), (0,1), (1,0), (1,1)) (see Kollar & Roy, 2009). For our purposes, this matrix is calculated by scene type in order to decrease the size of the pgm whose number of nodes and edges correspond with the number of possible objects in the given database. By querying scene type rather than the whole database, the size of the network becomes more manageable for performing the several inferences the predict the fixation sequence.

compatibility:

Calculates the energy function of the pgm based on the above cooccurrence matrix.

create_graph:

Uses the pgmax library to build the underlying probabilistic graph on which inference will be calculated. This function uses the compatibility function to calculate the energy function used for inference.

set_evidence:

Builds a list of the evidence (or associated probability) that a given variable (or object in this case) is likely to take on the value for a given state (0 or 1, absent or present). The evidence is therefore a matrix with the number of rows equal to the number of objects in the database and the number of columns corresponding with the number of states. This is then fed to the inference algorithm.

certainty:

Calculates the value of the evidence for each state (0,1) for each variable (object) that is fed to the **set_evidence** function. This certainty or likelihood is calculated as a function of the object's distance from the current fixation. The highest likelihood for each object is maintained during search such that if you fixate an object or very near it, the information about that object's state (presence or absence) is maintained across fixations.

get_centroids:

Creates lists of the set of objects and their centroids for a given image.

create_scene:

Creates a data frame to represent the latent scene information, i.e. the ground truth state information for each object within each patch of the image. Also creates a data frame for the observed scene which stores the likelihoods (certainties) and marginals for each patch. The patch size in pixels is a parameter to this function.

choose_target:

Creates list of potential targets based on the targets present in the ground truth image that aren't also part of the baseline objects for the pgm. The baseline objects for the pgm are calculated as the objects that are decoded as present in the default mode of the network. These tend to correspond with the objects that are consistently present across images in the database.

get_fixation_seq:

Main algorithm which calculates the fixation sequence for a given image using the pgm based on object cooccurrences which works as follows: The first fixation is starts at the center of the image. Inference (loopy belief propagation) is calculated at this patch. If the MAP state for the target is 1, then the target is found and search terminates. If the target is not found, the evidence is calculated for all patches based on their distance from the fixated patch. Inference is then performed for all patches and the patch with he highest marginal likelihood for the target is fixated next. This loop continues until a patch is fixated with a MAP of 1 for the target object or the maximum number of fixations is reached.

visualize_search:

Plots and saves the scene image with the grid of patches, the fixation sequence, and target listed.

preliminary results

Baseline model:

The baseline model for the pgm provides a default mode of the graph given the object cooccurrences for a particular scene type. Here are the baseline models for a few scene types:

Kitchen: ceiling, floor, sink, stove

Bathroom: bathtub, cabinet, floor, mirror, sink, toilet

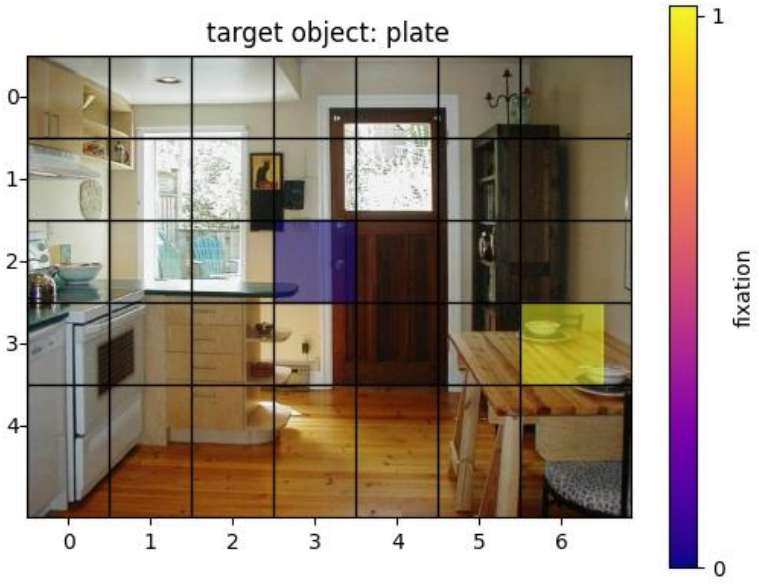
Bedroom: bed, ceiling, floor

Living room: ceiling, coffee table, floor, sofa

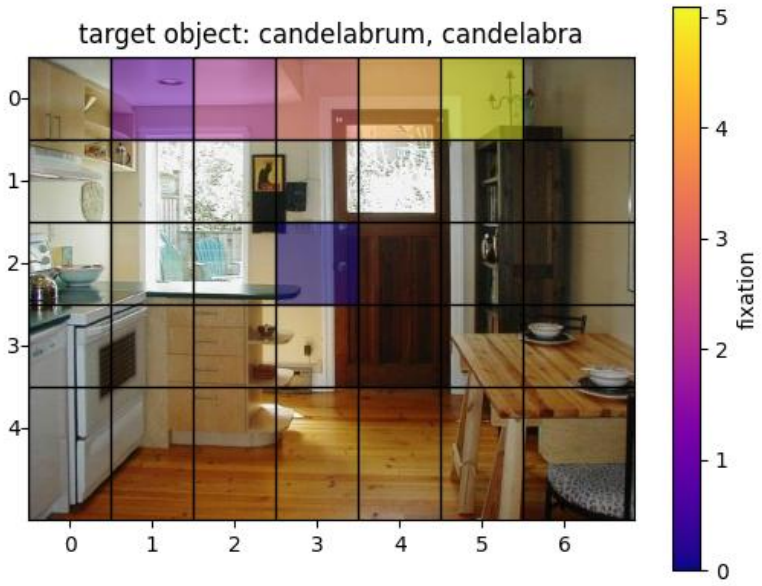
Fixation sequences:

The following fixation sequences demonstrate that this model accurately locates the patch for a given target.

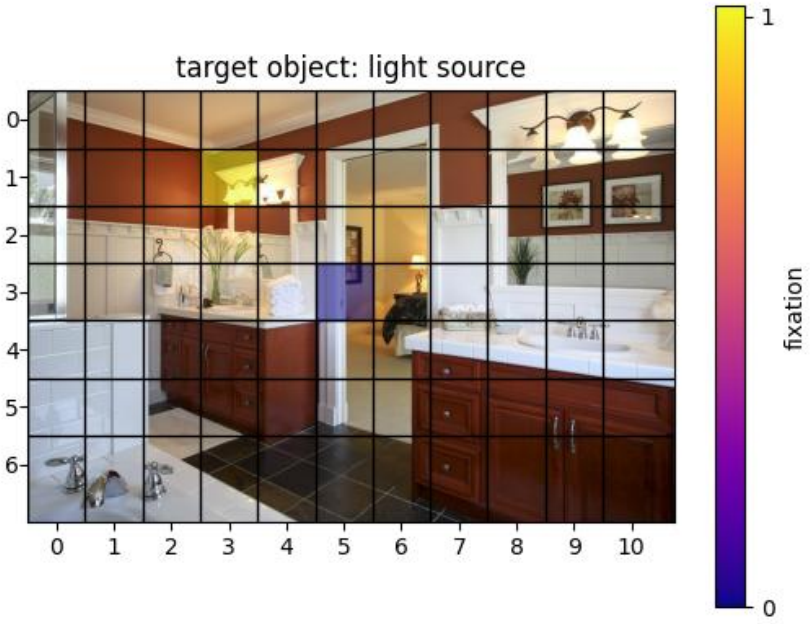
Kitchen (sigma = 2):



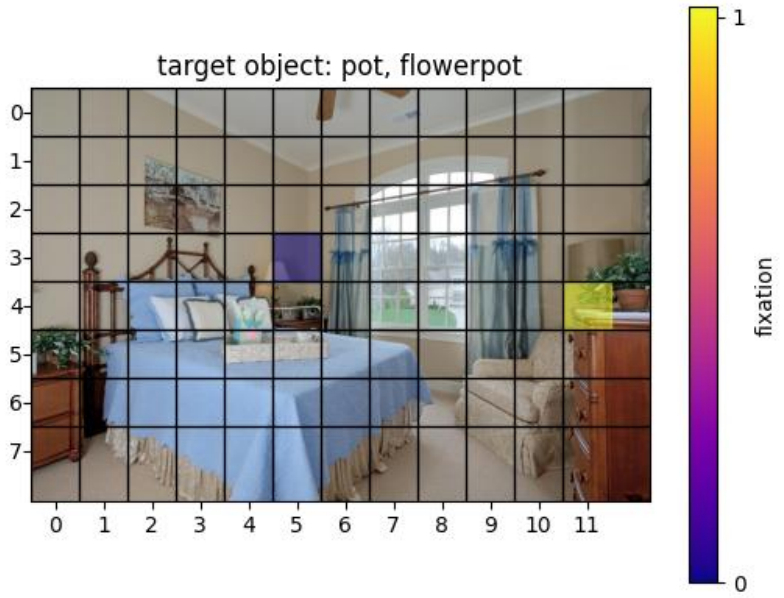
Kitchen (sigma = 1):



Bathroom (sigma = 3):

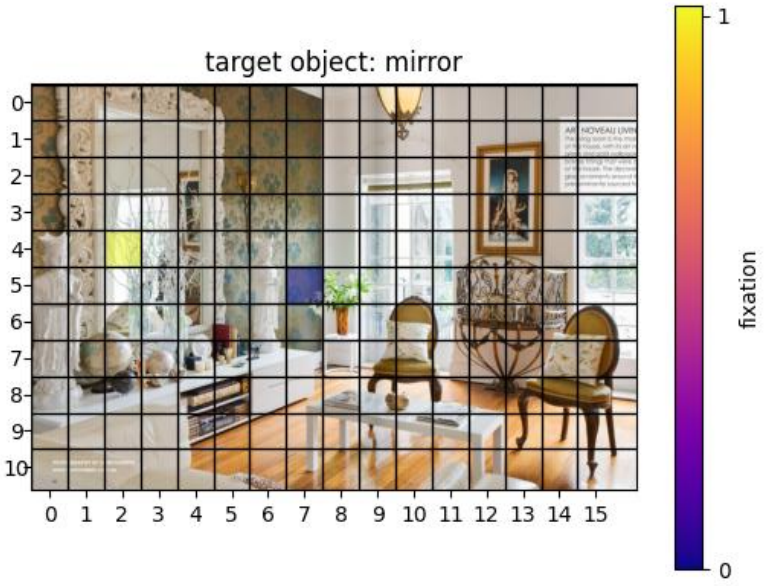


Bedroom (sigma = 3):

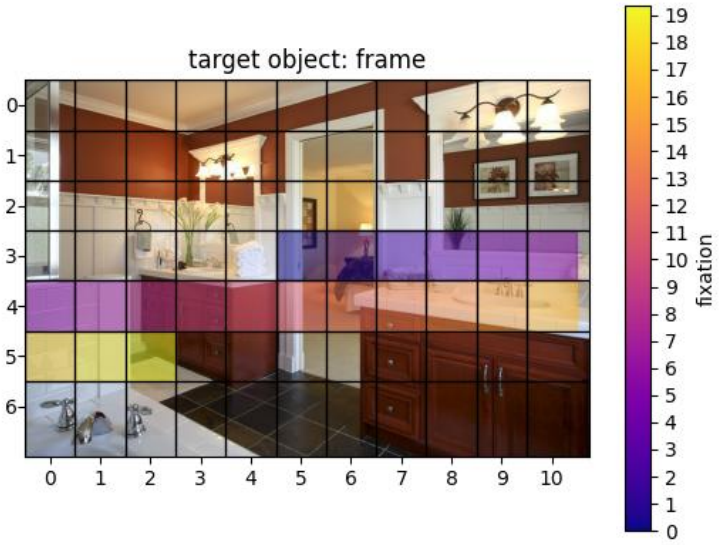


Bedroom (sigma = 2):

Living room (sigma = 4)



Example trajectory when sigma is too small (spatial scale is limited only to the fixated patch):



next steps

1. Calculate which objects fall into each patch based on pixel-wise object labeling rather than object centroids (i.e., even though the centroid of an object doesn't fall into a patch, some of the object might still be visible in that patch).
2. Appropriate sigma for fall off of evidence with distance from fixation. This will probably depend on this distance of the observer and thus the visual angle of a given area in the scene.
3. Appropriate calculation of evidence values based on distance.
4. Formal analyses of model accuracy (e.g. number of fixation sequences that reach the target, by how many fixations, across scene types).
5. Calculate object cooccurrences based on another dataset (e.g., NYU Depth V2, SUNRGBD, LVIS)
6. Combine local and global inferences using object occurrences on a smaller spatial scale and global on a larger spatial scale. The global cooccurrences would be intended to capture the larger spatial structure of scenes (i.e. where surfaces tend to be located and how the potential targets are associated with these surfaces [ref]). Separate cooccurrence matrices could be calculate for the local and global pgms by calculating cooccurrences by smaller and larger regions in the image rather than by the whole image alone, which is the current method.
7. Compare baseline models for local and global pgm to understand if surfaces versus local objects fall out of the default.
8. Test the model against a dataset of human fixation sequences collected using BubbleView and/or in virtual reality.