Towards Global Localization of an Indoor Mobile Robot using a Pre-Trained Neural Network

Mahim Agarwal, *31345405*

***Abstract*—To navigate reliably in an indoor environment, a mobile robot must have a constant idea of its position. Mobile robot global localization is the problem of determining a robots pose in an environment by using input visual data, when the initial position is unknown. This usually happens in two scenarios: the kidnapped robot problem where the robot is instantly moved to other position without being told during the operation of the robot, and the wake-up robot problem where it is carried to an arbitrary location and asked to start functioning. To solve this problem, Monte Carlo Localizations (particle filter approach) have been used previously but it gets more and more complex as the size of our map grows. So, to reduce the computation, we explored the use of pre-trained network to calculate an initial prior of robot's high level location estimate. This prior can then be used with particle filter approach to distribute the set of hypothesis points on a localized space rather than all over the map, thus reducing computation. The main idea of using pre-trained network is by matching distinctive features in the current frame with an already existing feature database of training images. The image features are extracted using 3 intermediate layers of a pre-trained AlexNet network. The proposed method is useful to solve problems in rescue robotics, where initial position is difficult to specify and robot incur constant perturbations which changes its position.**

I. Introduction

Every mobile robot needs the fundamental ability to navigate the environment. Two key problems in mobile robot navigation are global position estimation and local position tracking. Global position estimation is the ability to decide its position in a priori or previously learned map, given no other information than that the robot is somewhere on the map. If no priori map is available, many applications allow for such a map to be built over time as the robot explores its environment. Once a robot has been localized in the map, local tracking is the problem of keeping track of that position over time by compensating the incremental errors through odometry data. Both these capabilities are necessary to enable a robot to execute useful tasks, such as office delivery or providing tours to museum visitors. By knowing its global position, the robot can make use of the existing maps, which allows it to plan and navigate reliably in complex environments. Accurate local tracking on the other hand, is useful for efficient navigation and local manipulation tasks. Both these sub-problems are of fundamental importance to building truly autonomous robots.

This is a class project for CMPSCI 670: Computer Vision presented as a part of coursework requirement.
Mahim Agawral, University of Massachusetts Amherst.
Date: 12/20/2017

We adopt the view that probabilistic approaches are among the most promising candidates to providing a comprehensive and real-time solution to the robot global position estimation. In the particle filter based localization, the algorithm estimates the position and orientation of a robot as it moves and visualizes the environment. The algorithm uses a particle filter to represent the distribution of likely states, with each particle representing a possible state, i.e., a hypothesis of where the robot is. It starts with a uniform random distribution of particles over the configuration space, meaning the robot has no information about where it is and assumes it is equally likely to be at any point in space. Whenever the robot moves, it records its sensor data and updates the particle distribution so that more particles are concentrated on positions that will give similar sensor recordings. But, for large maps, this approach is very expensive as we need a lot of particles to be distributed over entire map. Also, to get a good convergence of these particles at a particular location, we will have to take large number of sensor readings. Instead, what we can do is supply the robot with an initial estimate of its location over an area or a room(high-level). This will allow us to start with already concentrated particles at our prior location, in the particle filter approach and will save subsequent computations of convergence. The initial prior is estimated as soon as the robot goes live and captures the first image. As it captures a new picture frame, it extracts features from the image and computes similarity of these features with the training features contained in a model database. The model database has feature vectors and the area labels that tell the robot what features belongs to which area. According to the similarity of the features measured, the robot extracts the k-closest features and re-distributes the particles giving more weight to the most similar area-feature and less to others. The hypothesis is that when this is repeated fairly small number of times, the particles will, ultimately, converge towards the actual position of the robot. After knowing the initial position, we can use established localization techniques to keep track of robot movement for future. And in case of localization failures, the global localization algorithm can be used again to get the current position.

The feature matching approach also provides an advantage over standard classification neural network by making it more scalable to train for new datasets. To train the robot for a new area, we can extract image features by running only the new dataset through the convolutional neural network and appending the results to our model database. This is unlike the case of using classification network where we will have to re-train the network over the complete dataset comprising of

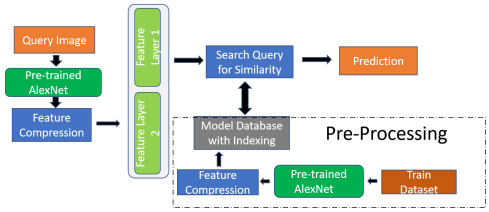

Fig. 1: Method Overview Source: [1]

old and new training examples.

II. Related Work

This work is largely based on and an extension of Eric Wilkinson and Takeshi Takahashi's work on efficient aspect object models using pre-trained convolutional neural networks [1]. They represented objects as collection of aspect features which are independent of the viewpoint. The classification problem was solved by comparing live feature aspects with the trained aspect database and predicting the most similar class and orientation of object. The model database was updated online if, within a class, the test aspect failed to meet the threshold similarity with training database. They compared several compression techniques to get an optimum run time for live robot test. The same methods are implemented for localization and room classification problems presented in this work.

Also, Monte-Carlo based particle filter localization is widely researched topic in robotics and this work is primarily a test of concept of previous findings and might represent many conceptual similarities with previously presented works. However, all the works referenced are duly cited and an attempt has been made to introduce novelty to make these techniques more efficient.

III. Method Overview

The method overview is pictorially represented in Fig. 1. We first create a model database by processing the training images through the pre-trained AlexNet[2]. The feature vectors Y of these images were extracted from 3 layers of CNN: final fully connected layer: fc7, penultimate fully connected layer: fc6 and maxpooled layer after final convolutional layer: pool5. These features are then compressed into smaller dimensional space and indexed into a tree before being stored into our model database. The model database is made up of tuples $\{Y_i\ , R_i, A_i\}$ where R_i is the room label and A_i is the area withing the room for that particular image.

When the robot goes live, the input to the system is a new image frame containing information about robot's current position. The image is scaled to the input image size of CNN and passed through the network to extract features at all 3 layers mentioned above. We again compress the features and get the input feature vectors for all 3 layers, Y_i. For each unique class identifier $C_i : (R_i, A_i)$ in the database, we precompute the statistical distribution of all Y_i stored with that class by fitting a Gaussian model G_{C_i}. This model is used to score the likelihood of input Y belonging to class C_i. In order to identify the current location using the model database, the input class vector Y is queried against the database so that the k closest tuples are returned as the result set, R. For each unique C_j appearing in R the system evaluates the likelihood of the input Y belonging to C_j using G_{C_j}. This is performed for feature vectors from all 3 layers for their different weighted combinations to identify the highest probable location of the robot.

IV. Experiment

To determine the optimum method for implementing localization on a real time robot, we considered 3 main hyperparameters and compared the performance of robot over them. The first one is choosing different compression algorithms based on the memory space they require and the query time needed to extract the most similar features given a test image. This comparison was made on only half of the data set within the constraint of time. The best performing compression algorithm was picked and used for whole data set.

The second hyperparameter was to determine the best performing similarity measure for finding co-relation between test and train image feature vectors. The performance was compared on the number of correct prediction the model can make for a given test image.

The third and the final hyper-parameter was to try different weighted combinations of features of 3 layers and determine the ideal layer(or combination) to extract features from for localization task.

A. Training Dataset

The training dataset consists of 1000 images of different areas within 7 different rooms of LGRC, UMass. The images were captured using mobile camera and were scaled down before creating the model database. The model can be easily extended to include more rooms and locations owing to the method used but the findings here are representative of this small sample space because of limited hardware resources and time constraints.

B. Compression Setup

Due to the time constraint in robotics systems and memory consumed by training feature vectors during localization, we want to compress feature such that:

- Query time for a new test image is less while it should return at least one vector corresponding to the test image area label.
- The memory consumed to load the model database is as low as possible.

To reduce query times and handle scalability, we used a Ball Tree indexed structure to store model database that efficiently performs nearest neighbor searches even for high dimensional data. [3] The query time and memory consumption was calculated for all 3 layers of CNN and then averaged over all 3 layers for comparison of the optimum algorithm. The parameters will also depend on the dimensions of the compressed

Algorithm	Dimensions	Memory Space*	Time(s)
IPCA	128	27100	0.026
KPCA	128	218000	0.031
SRP	128	3700	0.025
GRP	128	53400	0.026
Uncompressed	4096**	480000	NA
* in KBs(also include feature transformation vector size to transform test features to reduced dimensions)			
** 4096 or more features depending on the layers			

TABLE I: Performance comparison of different compression algorithms

space used and thus, for comparison, the dimensionality of the reduced feature vectors is kept constant for all the algorithms. The different compression algorithm used are: Iterative PCA[4], Kernel PCA[5][6], Sparse Random Projections[7] and Gaussian Random Projections[8].

C. Compression Comparison

Given query image I with class label (R, A) we measure the image retrieval performance of a compression algorithm whose result set is R over all images in the test set(50% of complete dataset). Since, the compression is only a one time thing, we will only measure: if it returns the correct index when tested, time it takes to query the k closest images and to calculate the similarity measures and its volume or the memory space required to store it. We ran all compression algorithms for reduction of feature vectors to 128 dimensions and the result are obtained as shown in Table I.

In terms of accuracy, all 4 algorithms performed equally good. Thus, Sparse Random Projection compression algorithm was selected to carry out the localization task.

D. Indexing

For indexing the large feature vectors in the model space, we used python's sklearn BallTree library. A ball tree is a space partitioning data structure for organizing points in a multi-dimensional space by partitioning data points into a nested set of hyperspheres known as "balls". It is very effective for performing nearest neighbor searches.
In a ball tree, every node defines a D-dimensional hypersphere, or ball, containing a subset of the points to be searched. Each internal node of the tree partitions the data points into two disjoint sets which are associated with different balls. While the balls themselves may intersect, each point is assigned to one or the other ball in the partition according to its distance from the ball's center. Each leaf node in the tree defines a ball and enumerates all data points inside that ball.
Each node in the tree defines the smallest ball that contains all data points in its subtree. This gives rise to the useful property that, for a given test point t, the distance to any point in a ball B in the tree is greater than or equal to the distance from t to the ball. Formally:

$$D^B(t) = \begin{cases} \max(|t - B.pivot| - B.radius, D^{B.parent}) & if B \neq Root \\ \max(|t - B.pivot| - B.radius, 0) & if B = Root \end{cases}$$

where $D^B(t)$ is the minimum possible distance from any point in the ball B to some point t

E. Similarity Measure

After extracting the 10 closest neighbors for fc7 layer from ball tree(using the default minkowski distance metric), we used spearman correlation distances to find the similarity between test and train features. This was done to improve upon the referenced work of Eric and Takeshi for object aspect classification problem. They used pearson correlation distance but spearman was found to work better for localization task.

Pearson's correlation is given by:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where, n is the sample size
x_i, y_i are single samples
and $\bar{x}, \bar{y}$ are means over all samples

And **Spearman's correlation** is defined as the Pearson's correlation coefficient between the ranked variables. So, for a sample size of n, the n raw scores of of X_i, Y_i are converted into ranks rgX_i, rgY_i and then computed by:

$$r_s = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X}\sigma_{rg_Y}}$$

where, cov is the covariance and σ is the standard deviation

The similarity measures are calculated over different weighted combinations of feature vectors layers and their performance was calculated over all the test images. The performance indicator was how well they identified a test image with the corresponding train image area. The results obtained are shown in the following section.

F. Results

The resulting images are attached on the next page(Fig 2, 3, 4, 5). These type of comparison were made on all the test images to check whether they give the correct Room and area predictors or not.

As we can see from the pictures, using only fc7 layer will not produce good result. It gives a relatively low score for correct predictions and comparable scores even for wrong predictions. Thus, we can never confidently say the exxact area the robot is located in.

From the 3^{rd} and 6^{th} weight combination results, for all images, it is clear that we should give more weightage to pool5 feature layer vectors since, they clearly differentiate between wrong and correct predictions on teh basis of magnitude of similarity scores. Also, more number of estimates for lower layer combinations are closer to the actual test image than that for high level feature layers.

Fig. 2: Test Image: corridor

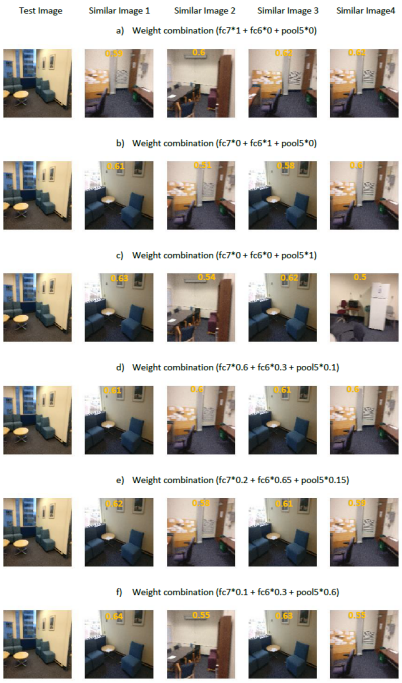

Fig. 3: Test Image: MS space

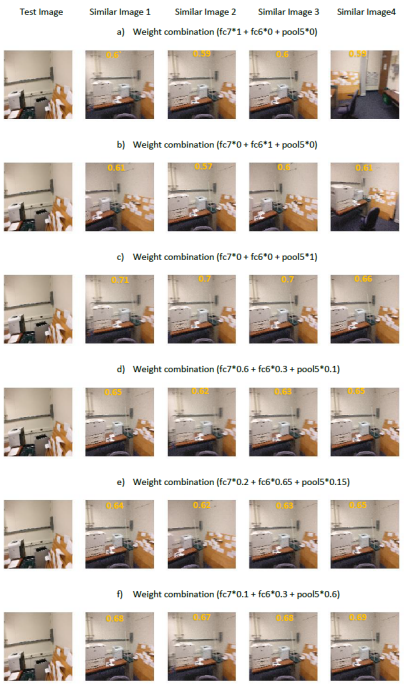

Fig. 4: Test Image: Printer room

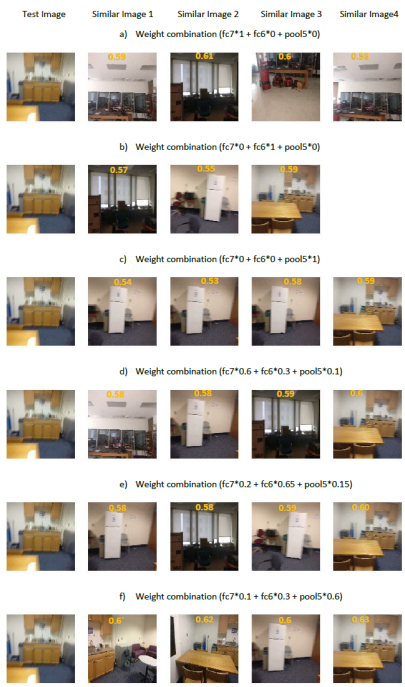

Fig. 5: Test Image: Kitchen

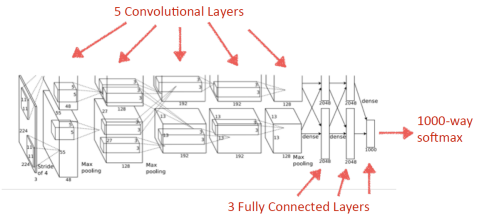

Fig. 6: AlexNet Architecture Source: [2]

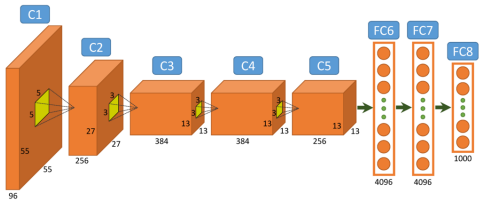

Fig. 7: AlexNet Sematics

V. Conclusion

It was found that the suggested feature matching approach can be reliably used to identify high level localization (like room and area). We are able to correctly classify the area in which robot is present and thus we can use this approach to give a prior for particle filter approach mentioned above.

Also, through experiment, we found that high level feature comparison are not enough to predict accurately and confidently and thus, more weightage should be given to pool5 or lower layers and less weights for high level layers in the alexnet network.

We also found that, for localization task, it is better to use spearman correlations distance (normalized for range 0 to 1) instead of pearson correalation distance used on object aspect classifier work being referenced here.

VI. Future Work

The next step is to use this prior estimate to initialize the particle filter distribution of Monte Carlo Localization method. And then use a map (through SLAM or other local localization methods) to determine the exact coordinate location of robot on the map.

Also, above results can be further improved by considering several other hyperparameters like leaf size of the tree, metric distance used for indexing, other similarity measures and combination of even lower layers of alexnet.

Appendices

A. AlexNet

The architecture is represented as Fig. 6 and Fig. 7.

References

[1] Eric Wilkinson and Takeshi Takahashi. "Efficient Aspect Object Models Using Pre-trained Convolutional Neural Networks". $ICHR15\_077\_FI$

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. 2012, pp. 10971105.

[3] Stephen Malvern Omohundro. Five balltree construction algorithms. International Computer Science Institute Berkeley, 1989

[4] Matej Artac, Matjaz Jogan, and Ales Leonardis. Incremental PCA for on-line visual learning and recognition. In: Pattern Recognition, 2002. Proceedings. 16th International Conference on. Vol. 3. IEEE. 2002,pp. 781784.

[5] S. Mika, B. Schlkopf, A.J. Smola, K.-R. Mller, M. Scholz, G. Rtsch, "Kernel PCA and De-Noising in Feature Spaces", Advances in Neural Information Processing Systems 11, pp. 536-542, 1999.

[6] C.J. Twining, C.J. Taylor, "Kernel Principal Component Analysis and the Construction of Non-Linear Active Shape Models", Proc. British Machine Vision Conf., pp. 23-32, 2001.

[7] Li, P., Hastie, T. J., Church, K. W. (2006, August). Very sparse random projections. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 287-296). ACM.

[8] Bingham, Ella, and Heikki Mannila. "Random projection in dimensionality reduction: applications to image and text data." Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2001.