Using Machine-learning Techniques to Increase the Efficiency of Kinect Fusion Reconstructions

Literature Review

Alan Lau

MComp Computer Science
University of Bath
October 2015

Contents

1 Introduction

This project aims to quicken the time required to create a 3-dimensional reconstruction of objects in a room with just a simple 2-dimensional scan from one angle, using a Microsoft Kinect Camera. To enable such goal, classification, a form of machine-learning, is proposed to be used. There are many different statistical classification algorithms that can be used for labelling objects through training with similar objects and recognising the category and object belongs when put in use.

Being an RGBD camera, not only does the Kinect Camera captures a colour (RGB) image, it also captures depth information via an infrared laser combined with a monochrome camera [1]. This information enables a detailed 3-dimensional reconstruction.

A group of New York University researchers have created a depth dataset called the *NYU Depth Dataset* [2], which is freely available online.[1] A variety of objects are scanned, labelled or segmented from a scene. The dataset is freely available for various applications in different formats. The number of scans for different classes of objects makes it ideal to be used as training data for the proposed classifier.

This Review attempts to explore in greater detail about the NYU Dataset and its applications, how depth maps are useful to 3-dimensional reconstruction and how the depth dataset is useful for training. We also attempt to justify the algorithms that will potentially fit to label objects in a reconstructed scene.

[1]The datasets are available for download at `http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html`

2 Kinect Fusion

The Kinect Camera was first released for the company's gaming console, XBox 360. It was designed to recognise gestures, faces and voices, providing a more physical way and new dimension to interact with the interface and games than a conventional controller. A Windows-compatible version of the Camera, an SDK and Kinect Fusion were released later, enabling researches and the development of commercial products [1].

Augmented reality (AR) and real-time reconstructions are some of the most popular research applications of Kinect Fusion. SemanticPaint [3] demonstrates the possibility of Kinect Fusion in AR in real-time scenes. It allows parts of the scene to be 'painted' by the user. It also uses segmentation and object recognition to paint similar objects in the same colour. Although this project does not involve real-time processes, it provides the knowledge required to create or use an existing segmentation algorithm for labelling individual items from a scene (discussed later).

2.1 Kinect Camera Technologies

The two generations of Kinect Camera use different 3-dimensional camera technologies to obtain depth information about a scene. Each of these technologies has its pros and cons, which is discussed below. However, the common problem of these technologies is that it does not deal with very bright light, where detail will be lost [4].

2.1.1 Structured Light

Structured light is used in the first generation Kinect Camera (2010). A sequence of known infrared patterns are projected onto the scene. A deformed pattern is formed when objects are 'in the way' of the patterns. The object is then observed from another angle by the monochrome camera. Through analysing the deformed patterns and observations, the depth information about the scene can be obtained [4][5]. It is worth noting that the NYU Dataset [2] uses information captured by the first generation of the Kinect Camera.

2.1.2 Time-of-Flight

Rather than looking at the deformed pattern, the second generation Kinect Camera (2013) estimates depth information based on the time the infrared beam takes to travel back and forth. The difference between the reference signal and returned signal allows the calculation of a time difference, which helps estimating the required depth information [4].

2.2 Data Representation

2.2.1 Point Cloud

A point cloud stores data points in a coordinate system. [4] In a real-world case (as well as the captured images), a 3-dimensional coordinate system is used. A point cloud can be used to generate a mesh so that it can be used to render a visual image of the reconstruction volume.

2.3 Reconstruction

2.3.1 Pipeline

A single raw capture with the Camera does not provide a detailed reconstruction. Combining the depth information of many images together enables a super-resolution [3d-paper] reconstruction, making a detailed and high quality reconstruction possible. The following is the full pipeline of how a reconstruction is created from raw depth data:

1. Raw Input Conversion
    - The raw *depth map* is captured by the infrared-monochrome subsystem of the Camera. This information is not very detailed.
    - It needs to be combined with the *normal map*, which is the surface normals associated with each vertex [6], to provide a more detailed but noisy reconstruction at this stage [7].
    - Further conversion and reconstruction is needed to retain detail, remove noise, and fill in holes missed by the Camera, possibly due to bright light.
    - This information is stored as a point cloud and will be combined into one representation later on.
2. Camera Pose Tracking
    - The location and orientation (world pose) of each frame are tracked as the Camera moves around.
    - This alignment is constantly traced, allowing all the point clouds to be aligned together [1].
    - The frames captured from different poses, even the smallest movement (e.g. caused by a hand-shake), will allow further quality improvements to the scene, achieving more than what a single raw capture is capable of [8].
3. Fusing
    - The depth data converted from the raw input is combined into a single 3-dimensional space per frame.
    - A running average of depth is kept. This reduces noise, and creates a refined reconstruction by combining all the information in one place [1] [8].
4. Resultant Reconstruction Volume
    - The resultant point cloud will be of a highly detailed reconstruction of the scene. For example, grills of a millimetre [**?**] can be reconstructed properly.

- A rendered image of the 3D reconstruction volume is possible by using methods such as ray-casting, providing a visual feedback of the scene, and allows for many possibilities, such as augmented reality applications.

2.3.2 Volumetric Reconstruction

By averaging the surface models and depth data from multiple viewpoints into one volumetric voxel volume, the scene appears to live in a 3-dimensional 'box'. This provides a three-dimensional view of the scene, allowing more data to be used

TODO: describe how Microsoft Research [8] creates a volumetric reconstruction

2.3.3 Surface Reconstruction

TODO: describe how Microsoft Research [9] does surface reconstruction

2.4 Segmentation

In order to identify an object in a scene with many objects, there needs to be a way to single them out individually so that operations such as labelling can be done on them. In computer vision, this is called *segmentation.* There are many segmentation algorithms available. Some popular ones include *k-means*, *Gaussian*, *mean-shift*, *normalised cuts*, *similarity graph-based segmentation* and *binary Markov random fields using graph cuts.*

TODO: discuss in brief details about some of these algorithms - especially k-means and mean-shift, as they are the popular models

TODO: segmentation using mean-shift by the creators of the NYU Dataset using the dataset [2]

TODO: segmentation using "an efficient mean-field inference engine applied to a dynamic conditional random field model" [3]

3 Classification

The idea of classification is to identify of which group an object belongs to. This relies heavily on a good algorithm that is able to group similar objects together in the first place. In more formal words, a classifier groups objects that has some semantic similarity enough to be classed as the same type. Each class has a label in which these objects are identified as. For example, round objects that can hold liquid can be classed as "bowls", despite being different in colour or and of slightly different shapes [10].

3.1 Classification Types

There are many classification models that are fit for different purposes. They are broadly split into two groups - *supervised* and *unsupervised (clustering).* The following explains what they are and some associated algorithms that could be used for this project's classifier basis.

3.1.1 Supervised Classification

Supervised learning uses pre-labelled data to train the classifier. Correctly labelled data is given to the classifier at training. This means that the labels and the number of them are pre-defined so that the classifier has a limited amount of choice. The classifier has to aggregate and understand the similarities so to be able to say which class an unknown object is, based on what is already known. [11]

A supervised classifier can either be *probabilistic-based* and *geometric-based.* Probabilistic-based algorithms involves a probabilistic density function (Gaussian distribution being one of the most well-known ones), and are sub-divided into *parametric* and *non-parametric.* For a parametric classifier, the statistical probability distribution of each class is known and is used, whereas the purpose of a non-parametric classifier is to estimate the distribution, as the number of parameters is not known, or does not matter [11] [10].

Supervised learning is ideal when there are an abundance of correctly labelled data for the classifier to learn from. Figure 3.1 describes this graphically. A *training set* contains some pre-labelled data which the classifier algorithm will be trained on. Some of the pre-labelled data is set aside and used as the *test set* to see how well the trained classifier performs.

3.1.2 Unsupervised Classification

Unsupervised classification is also known as clustering. Items with similar properties in the data level will be grouped together by the algorithm itself. Training is still required, otherwise the classifier will not be able to put a label to an unknown item. The training data should be unlabelled, to allow the algorithm decide on the number of clusters (labels). Ultimately, the goal with unsupervised learning is to avoid any intervention and let the algorithm does its job [10].

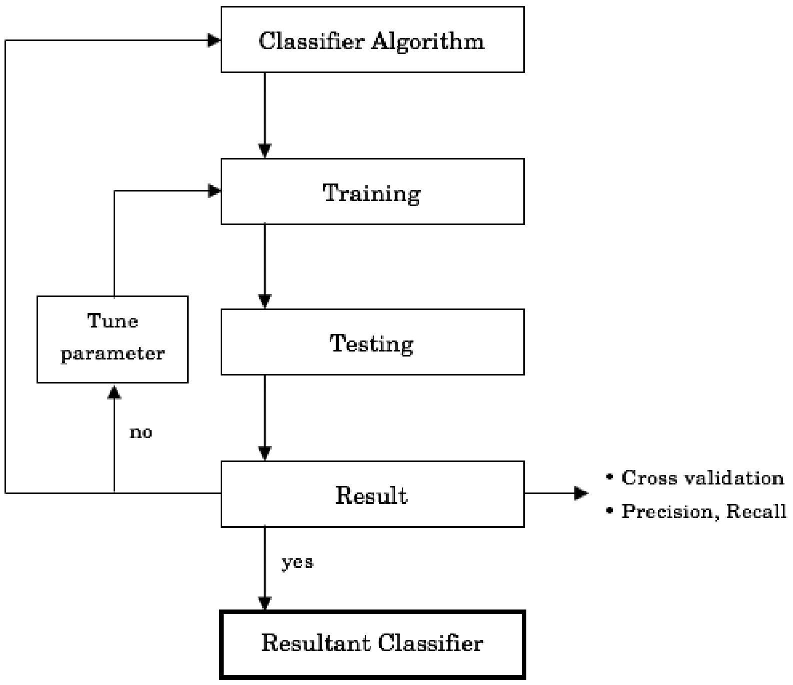

Figure 3.1: An overview of the process for getting a supervised classifier [11]

This should not be confused with *online learning.* An unsupervised classifier does not

3.1.3 NYU Depth Dataset V2

Quality training data is required to achieve high precision and accuracy. The NYU Depth Dataset provides a quality wide range set of depth data which can be used for training a classifier with the aim of achieving high precision and accuracy. There are 1449 densely labelled and 407 024 unlabelled newly scenes in this version. The large amount of wide variety of scenes and items of the same class is ideal to be used as training data [2].

The abundance of available data means that time can be saved from having to locate many objects of the same in different scene settings, but effort can be focussed in getting the *best*

3.2 Machine-learning in Python

3.2.1 `scikit-learn`

`scikit-learn` is a machine-learning library for Python, which is actively in development. It has quality implementations of popular machine-learning algorithms for different machine-learning problems. It is built on `numpy` and `scipy`, making it easy for data manipulation [12]. A wide range of companies and researches use scikit learn for machine-learning purposes, including Spotify and Evernote.[1] Pedregosa et al. [12] copmared it with other Python machine-learning packages which sees scikit-learn beating

[1]`http://scikit-learn.org/stable/testimonials/testimonials.html`

the others in baseline tests.

Supported supervised classification algorithms include Support Vector Machines (SVM), Random Forest, Stochastic Gradient Descent, Nearest Neighbours and Decision Tree.

3.2.1.1 R, MatLab

incomplete...

3.3 Performance

3.3.1 Precision and Recall

The precision and recall rates (accuracy) can be calculated to analyse the performance of the classifier. [13] They are calculated by comparing predictions against the actual values.

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

Table 3.1: Confusion matrix (taken from [13])

In order to calculate these rates, a confusion matrix is computed. It is a table that puts the correctness of predictions against the actual values of the underlying data (Table 3.1). This shows the number of correct and incorrect recognition, allowing a couple of metrics to be calculated, notably precision and recall. (Table 3.2)

$$Recall = \frac{TP}{TP+FN}$$

$$Precision = \frac{TP}{TP+FP}$$

$$TruePositiveRate = \frac{TP}{TP+FN}$$

$$FalsePositiveRate = \frac{FP}{FP+TN}$$

Table 3.2: Common machine-learning evaluation metrics (taken from [13])

Precision can be described as *"out of the predicted positives/ negatives, how much did the prediction is correct"* , whereas recall is described as *"out of the actual positives/ negatives, how much did the prediction is correct"*, as seen in Table 3.2. The former tells us how well the model can predict correctly; the latter tells us the *quantity* of which the relevant items is picked. [14]

To say that a model is *good for use*, it has to be both precise and has a high recall rate. If a model is very precise, but does not pick up much of the relevant items, the model might look as if it was performing well, but in fact, it hardly picked up enough of the right items; vice versa.

3.3.2 Cross-validation

It is important to ensure that the training and testing sets are different and that there is no overlap between the two sets, otherwise overfitting will happen. Overfitting is evident when a classifier appears to be functioning perfectly during the train-and-test stage, but turns out to be unable to classify new objects when put into use. On top of the training and test sets, a third validation subset can be introduced. This enables an extra check to ensure that the classifier actually works. However, the precision and accuracy of the classifer could suffer as a result of a smaller training set [15].

Cross-validation solves the problem by creating some *mutually exclusive* folds (subsets). These folds are created after a test set is taken out from the original set of data. One popular algorithm is k-fold cross-validation. It does not run validation on all possible splits on the folds but one different split for each run to save on computational complexity. Multiple cross-validation runs with different splits are performed, resulting in an average that estimates the complete cross-validation [16]. This gives an unbiased view as to how the classifier performs, as it is not limited to one fixed set of testing data.

3.3.3 Statistical Testings

TG Dietterich [17] proposed five statistical t-tests that can be used to compare different supervised classification algorithms. Statistical testings provide a good way to unbiasedly compare different algorithms and see how they actually perform in a particular situation.

TODO: read the paper in detail and find other relevant sources about other statistical tests that could be used[17]

3.4 Classification Algorithms

In this section, we focus on what scikit-learn can offer with its implementations of some of the most popular algorithms.

3.4.1 Notable Supervised Algorithms

`scikit-learn` supports many supervised algorithms and

- **Stochastic Gradient Descent (SGD)** fits a linear model efficiently and is particularly useful when there is a wide and large dataset for training.
- **Support Vector Machines (SVM)** enabels multi-class classification with different algorithms based on the nature of the kernel. Some examples are linear SVM and RBF SVM (1.4 [18].
- **Nearest Neighbours (NN)** has a number of implementation with `scikit-learn`. In general, works well with larger datasets. It could reduce noise but blur the boundaries of the clusters, making it harder to distinguish between labels (1.6.2 [?]).
- `scikit-learn` uses an implementation of the CART **Decision Tree (DT)**. It extends C4.5, a well-regarded implementation with high accuracy while adding support to regression and other features (1.10.6 [18]).
- **Random Forest (RF)** has many trees. They are built by the best split within a random subset. The result is based on the average of these predictions (1.11.2 [18]).

Model	Calibration	Training Speed	Fast at Classifying	Mean Accuracy
BST-DT	PLT	Slower than DT	?	0.917
RF	PLT	Faster than SVM	?	0.898
SVM	-	Slow	?	0.880
DT	ISO	YES	?	0.774

Table 3.3: Comparing properties of different classification models (accuracy information from [19]

Table 3.3 shows some of the empirical comparison results done by Caruana and Niculescu-Mizil [19]. 11 binary tests are performed and their averages of the outcome form the accuracy rates for each of the classifiers. As different algorithms have different characteristics and are fit for different purposes, calibration with algorithms such as isotonic regression aims to provide an unbiased view on how well the algorithms perform.

It shows that boosted tree performed the best while Random Forest performed better than SVM. There is some discrepancy between this and `scikit-learn`'s documentation [20]. The documentation shows that the SVM implementations in `scikit-learn` has the best performance.

SVM is known to not scale well and train slowly. If the results provided in [19] is trustworthy, there is little incentive to use a slow algorithm that does not boost precision and accuracy. It is also known that a Linear SVM, which is used by [19] does not perform well if the clusters are closely packed together. This could explain the discrepancy.

3.4.2 Notable Unsupervised Algorithms

TODO: describe and compare k-means, mean-shift, DBSCAN

3.5 Using Classifiers for Depth Data

TODO: Discuss how Naive Bayes, Decision Trees and SVMs are used in the Forestry Robot paper [21]

TODO: segmentation using online learning with a real-time random forest algorithm in Semantic Paint [3]

4 The Project

When creating the classifier, it is important to conduct many tests to ensure it is performing properly and well. As discussed, precision and recall, cross-validation and statistical testing are helpful tools to help achieve a good should also be done to ensure that the classifier performs properly based on an unbiased view.

It is reckoned that the performance at classifying a scene is more important than the time required at training. The trade-off between training time and precision and recall is little, so it is worth trying Gradient Boosting Classier (Boosted Tree), Random Forest and SVM to try to obtain the highest accuracy and low false positive/negative rates. Simple Decision Trees seem to perform less well compared to the others.

TODO: discuss if using multiple classifiers and combining the results like [21] should be approached

TODO: discuss how to use segmentation methods described by Semantic Paint [3] and the NYU Dataset research [2].

Bibliography

[1] Microsoft Coporation. Kinect Fusion MSDN Documentation.

[2] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*, 2012.

[3] J. Valentin, V. Vineet, M.-M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Niessner, A. Criminisi, S. Izadi, and P. Torr. Semanticpaint: Interactive 3d labeling and learning at your fingertips. *ACM Trans. on Graphics (TOG)*, August 2015.

[4] Ling Shao, Jungong Han, Pushmeet Kohli, and Zhengyou Zhang. *Computer Vision and Machine Learning with RGB-D Sensors*. Springer, 2014.

[5] Hamed Sarbolandi, Damien Lefloch, and Andreas Kolb. Kinect range sensing: Structured-light versus time-of-flight kinect. *CoRR*, abs/1505.05459, 2015.

[6] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[7] Microsoft Coporation. Kinect Fusion Project Page.

[8] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[9] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE ISMAR*. IEEE, October 2011.

[10] Peter Hall. Fundamentals of Pattern Analysis Notes. 2015.

[11] G Sahoo et al. Analysis of parametric & non parametric classifiers for classification technique using WEKA. *International Journal of Information Technology and Computer Science (IJITCS)*, 4(7):43–49, July 2012.

[12] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011.

[13] Jesse Davis and Mark Goadrich. The Relationship Between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML ’06, pages 233–240, New York, NY, USA, 2006. ACM.

[14] David Martin Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011.

[15] scikit-learn. Cross-validation: evaluating estimator performance (Documentation).

[16] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

[17] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

[18] scikit-learn. scikit-learn: Supervised learning (Documentation).

[19] Rich Caruana and Alexandru Niculescu-Mizil. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 161–168, New York, NY, USA, 2006. ACM.

[20] scikit-learn. Classifier comparison.

[21] Mostafa Pordel, Thomas Hellström, and Ahmad Ostovar. Integrating kinect depth data with a stochastic object classification framework for forestry robots. In *ICINCO (2)*, pages 314–320, 2012.