Facial Re-Recognition with OpenCV and Neo4j Graph Databases

Jared Beekman
McNeese State University
Lake Charles, LA
msu-jbeekman@student.mcneese.edu

ABSTRACT

In this paper, we will examine the problem of facial re-recognition, storage, and retrieval in order to refine of source images and identify subjects in subsequent images. The OpenCV computer vision library is used as the decomposition and recognition engine by performing Eigenface, Fischerface, and Lesser Binary Pattern decompositions in order to reduce storage requirements and computational overhead. These decompositions are subsequently stored, with the original source image, into a Neo4j graph database to preserve, identify, and generate relationship information either known to the end user or inferred from presence in multi-subject images. Upon the introduction of new images the system will then perform the three decompositions on all found faces generating relationship data from the source image to those decompositions. Subsequent queries such as "is this person contained in this orphaned image," or "since this person is in this image, are there any of their connected persons in the image" can later be executed more efficiently by amortizing computation and only comparing previously generated decompositions. Potential applications, and the equally important, ethical considerations of such a system will also be discussed.

I. INTRODUCTION

The facial recognition system we will discuss has at its foundations three methods of image deconstruction implemented in OpenCV. These algorithms are motivated by the fact that an image lies in an $\boldsymbol{m} \times \boldsymbol{n}$ dimensional vector space, where $\boldsymbol{m}$, $\boldsymbol{n}$ are the row and column widths of the matrix representation of the pixel data. As such, even a very small 100 x 100 grayscale image lies in 10,000 dimensions. The question then is "is all of this information useful?" The answer to this question, as in most statistical based questions, is that it depends upon the amount of variance in the data. Thus the regions with the most variance become our regions of interest as they will account for a vast majority of the information. [1]

II. EIGENFACES ALGORITHM

Principal Component Analysis (PCA), which forms the heart of the Eigenfaces algorithm OpenCV implements, calculates the eigenvalues and eigenvectors of the covariance matrix and the principal components are then the $\boldsymbol{k}$ largest eigenvalues. Then Eigenfaces uses the fact that given an $\boldsymbol{m \times n}$ matrix with $\boldsymbol{m > n}$ there can only be $\boldsymbol{n-1}$ non-zero eigenvalues to further refine otherwise, even a modest problem involving 400, 100 x 100 pixel images results in roughly

0.8GB of data. Efficiency aside, the Eigenfaces algorithm works by placing all the training samples onto the PCA subspace, projects the query image onto the same subspace, and then performs a nearest neighbor search between the training and query images. One of the drawbacks of the Eigenfaces method is that along with facial feature data it also encodes illumination, and while this information can be useful given a small set of training data which is only obtained under ideal, or singular, lighting scenarios can make re-recognition very difficult in subsequent queries, especially if the primary cause of the variance in PCA analysis is not facial features. [1]

III. Fischerfaces Algorithm

While PCA and Eigenfaces are powerful they do not consider any classes of information, and as such, information may be lost when components are ignored. Also, the components may contain no useful, for us, information at all as the variance could be caused by a non-facial source like lighting. The Fischerfaces algorithm is an application of Linear Discriminant Analysis, which was invented by Sir R. A. Fisher in 1936, which performs class-specific dimensionality reduction and groups the discovered classes together. A drawback to this approach is that it is heavily dependent upon the quality of the input data. If, for example, the system is never trained with poor illumination Fischerfaces is likely to find wrong components since it has never had a chance to learn what illumination does to the subject. [1]

IV. Local Binary Patterns Histograms (LBPH)

As noted previously the Eigenfaces and Fischerfaces algorithms have issues specifically with how the subject is lit. The general idea of LBPH is to summarize the area around a pixel and apply a binary value upon comparison with its center. The comparison is performed as if the neighboring pixels values are less than the target, then it gets assigned a zero, else it is assigned a one. After interpolation of all resultant circumference points not actually corresponding to points on the image the resulting deconstructed image is subsequently resilient against lighting differences. In order to perform recognition the deconstruction is then divided into a number of local regions whose histograms are concatenated for comparison. [1]

V. Graph Databases

Monumental amounts of image data in the form of pictures and videos are generated and stored. For example, in 2014 Instagram and YouTube users upload approximately 220,000 and 72 hours of new photographs and video content respectively [2]. Traditional relational database management systems (RDBMS) and NoSQL databases perform very fast, and efficient, queries on data associated with a single user, but fail miserably at querying information that involve recursive relationships such as those which form the structure of a social network. Experiments in this area displayed an exponential growth of execution time for an RDBMS system, whereas the graph database solution displayed logarithmic growth [3, Table 1]. As relationships can be identified, or inferred, from the presence of people in photographs and videos from known subjects we will be using Neo4j graph database engine to store our data.

Neo4j Graph Database Performance versus RDBMS System

Depth	RDBMS exec time (s)	GDB exec time (s)	Records returned
2	0.016	0.01	~2500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

Table 1: Graph Database versus RDBMS system runtime evaluation on database containing 1,000,000 people with an average of fifty relationships each [3].

VI. The Face Graph Deconstruction, Storage, and Re-Recognition Engine

The use of the system is divided into two parts. The first is that a user will be able to add people with a set of training data or an image containing one or more people to have faces extracted from. While the second is executing searches against the peoples deconstructions and images of unknown persons. The data store consists of a Neo4j graph database which treats people, saved training deconstructions, images, and grayscale sub images, which are further subdivided into grayscale copies of the original and grayscale cropping's of the faces contained within, and with the actual image and saved training files to be stored onto permanent media, which currently consists of the RPI's SD card, with path information stored in the nodes of the database. All coding for this project is in Python 2.7 using OpenCV 2.4.10 and Py2neo 2.0 libraries, and running on a Raspberry Pi 2 B with the Raspbian operating system. Py2neo provides object oriented structures for nodes and relationships, and access to the Cypher Query Language (CQL) for more complicated, manual interaction with the database. Further detail about the subsystems is provided in a sectioned approach.

VI.i. Neo4j Data Representation Model

Nodes and relationships in Neo4j contain an arbitrary number of key, value pairs such as name="Jared Beekman", age=40, or id=1, and a type description system. The type system is not like that of C, but is more descriptive in the sense that the type Person says that this node contains or represents person data. A CQL description of a node using our previously defined data could be (n:Person {name='Jared Beekman', age=40, id=1}), where 'n' is an optional reference name that could be used later in a query. A relationship between two people can then be modeled as (p1)-[r:KNOWS]->(p2), where 'r' is an optional reference name like 'n' above. Relationships are always directed, they are also implicitly bidirectional as direction can be ignored in CQL, and never dangling; i.e., (n)-[r]-> is invalid, but (n1)-[r]-(n2) is valid [5, 6].

VI.ii Adding a Subject to the Database

Adding a person to the database is more than just simply creating them a node. The generalized algorithm as is follows. Create nodes for the person, deconstruction, and all training images. Load all training images into memory. Scale, crop and rotate all training images to reduce the amount of background noise and assure congruent matrix size. Insert all node and relationship

data into the database, and save all files to the data store. Generate, train and store all deconstructions to the data store.

VI.III Adding a Group Photographs Faces to the Database

VI.IV Searching the Database

There are two ways in which to search this database. A brute force approach which takes all the people in the database and compares them to all images not yet classified which were inserted via the FaceRipper class, and upon addition of a new person to verify that they have not been added yet. The process is essentially the same, as only the source of the target data is different, and since the former was used in the experiment we will only cover that here.

For every person in the database load their stored deconstructions into memory. Get all images in the database which do not have edges to people. For each of those faces get the predicted label and confidence, aka distance, measurement from the recognizer. If this face is a match generate an edge from the person to the face and original images. Mark this face for later incorporation into the stored deconstruction which will further refine the deconstruction.

VII Experiment

VII.I Setup and Methodology

In the experiment the author selected twelve images in varying lighting scenarios covering a span of approximately four to five years these images form our training set. Also selected was an image containing two colleagues from approximately two years ago the faces from this image form our test set. As the Eigenfaces and LBPH algorithms are well tested the training set was not partitioned, and we can consider this experiment to be verification of our layer on top of OpenCV's algorithms. Fischerfaces was not done due to misunderstanding of the inputs to the recognizers which is further documented in results.

VII.I Execution

A person was generated and entered into the database with the training set on an empty database and, then the test set was likewise generated and stored. The file store and database contents subsequently verified correct through inspection and query. Then using the search function was called to compare the sole person and the training set to the sole test set.

VII.I Results

The FaceRipper class pulled four 'faces' from an image only containing three. Face ID 4 is a false positive of a group of items behind the actual subjects. This can easily be remedied by adjusting the parameters when doing face detection with the haarcascade classifier, but no such mechanism was put in place as of yet. Table 2 demonstrates a true positive for the Eigenface algorithm for Face ID 5, but a false negative with the LBPH algorithm for Face ID 5 and a false positive for Face ID 4. As previously noted, Fischerfaces was not ran during this experiment. The results therefore were decisively inconclusive, and revealed methodological, design and implementation errors.

Face ID	Eigenface Algorithm Confidence	LBPH Algorithm Confidence
6	10094.18	258.71
5	7479.20	222.14
4	8101.02	191.19
3	11148.14	272.27

Table 2: Confidence of match from Eigenface and LBPH algorithms and their corresponding face id numbers.

VIII. Potential Applications and Ethical Consideration

There are serious ethical considerations to consider before deploying a facial recognition system. Second generation bioinformatic systems such as these are characterized by their non-intrusive nature, unlike first generation such as fingerprints, DNA, etc. As such, the users of the system are only the owners and technicians whom have total control over others people's data. Data the people typically do not know they are giving away. Some system owners will display a sign at their entrance saying something to the effect of 'Warning Smart CCTV System in Use,' though the chances of the average person knowing what a smart CCTV system does is probably fairly slim. Also, the accuracy of facial recognition in uncontrolled systems is of high concern. Tampa, FL, and Virginia Beach, VA both installed facial recognition systems in busy neighborhoods to detect wanted criminals. Tampa even went so far as to scan the faces of 100,000 people at the 2001 Super Bowl. The threat of a system obtaining a false positive, identifying a non-criminal as criminal, is a very real threat. "Both Visionics (the company that manufactured Tampa's system) and Tampa police claimed, for example, that the chance of a false arrest is acceptable trade-off for the possibility of arresting a criminal who might otherwise remain at large" [4]. While this may seem logical it is a slippery slope indeed. During a thought exercise several members of our department wondered what could happen should this technology be abused. The answer was that one could sit outside the Jennings, LA Walmart, which is one of the only grocery stores in a relatively small city, with a hidden camera and inside of a few months not only map nearly 100% of the population, but use a frequency analysis and infer relationships between people without knowing anything about them. To summarize the risk the International Trade News quotes Edward Snowden as saying "Secondly, mass surveillance enables a capability called 'retroactive investigation' which more or less means that the government can access a complete record of your daily activity going back years – with no probable cause and without breaking the law. Snowden explained: 'You might not remember where you went to dinner on June 12th 2009, but the government does"' [7].

IX. Future Work

The future of this system lies largely in moving it onto a client-server architecture. The Raspberry Pi 2 B is more than powerful enough to work as both for a very small application, convenience store or something, if one had an external hard disk to prevent SD card corruption. However, Neo4j is an inherently distributed database system for scalability. For any application which would exceed the community editions limitations of 1,000 nodes and 5,000 relationships

there really needs to be an external database/file host involved. Work also needs to be done in order to remove the methodological errors in the design and implementation of the system, and a deeper understanding of the confidence numbers themselves must be undertaken.

REFERENCES

[1] "Face Recognition with OpenCV." *Docs.OpenCV.org*. OpenCV Documentation, Unk. Web. 10 Jan. 2015.

[2] Gunelius, Susan. "The Data Explosion in 2014 Minute by Minute Infographic." *ACI.info.* ACI Information Group, 12 Jul. 2014. Web. 27 Feb. 2015.

[3] Robinson, Ian with Jim Webber and Emil Eifrem. *Graph Databases*. Cambridge: O'Reilly, Unpublished 2015. Kindle file.

[4] Brey, Phillip. "Ethical Aspect of Facial Recognition Systems in Public Places." Journal of Information, Communications and Ethics in Society 2004: 97-109. Web. 28 Jan 2015. <http://www.utwente.nl/bms/wijsb/organization/brey/Publicaties_Brey/Brey_2004_Face-Recognition.pdf>.

[5] "Graph Academy." *Neo4j.com*. Neo Technology, Inc, Unk. Web. 14 Apr. 2015.

[6] Bachman, Michael. "Modelling Data in Neo4j: Bidirectional Relationships." *GraphAware Blog*. GraphAware, 11 Oct. 2013. Web. 14 Apr. 2015.

[7] "Mass Surveillance is an Invasion of Privacy." *InternationalTradeNews.com*. International Trade News. Unk. Web. 13 Apr. 2015.