DUPLICATE IMAGE DETECTION SYSTEM USING SPHERICAL LOCALITY SENSITIVE HASHING

SUDHA ANANDHI.N

11PCR19

A Project Report submitted to Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore in partial fulfillment of the requirements for the

Master's Degree in Computer Science

MAY 2013

DUPLCATE IMAGE DETECTION SYSTEM USING SPHERICAL LOCALITY SENSITIVE HASHING

SUDHA ANANDHI.N

11PCR19

A Project Report submitted to Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore in partial fulfillment of the requirements for the Master's Degree in Computer Science

MAY 2013

SIGNATURE OF THE

HEAD OF THE DEPARTMENT

SIGNATURE OF THE

SUPERVISOR

SIGNATURE OF THE

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I would like to express my sincere thanks to God Almighty, for His constant love and grace that He has showered upon me.

I am very grateful to **Dr.T.S.K.Meenakshi Sundaram, M.A., M.Phil., Ph.D., Chancellor,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore for his support and encouragement during the course of my study.

I heartily thank **Dr. (Mrs).Sheela Ramachandran M.Sc., P.G. Dip., Ph.D., Vice Chancellor,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore for extending all resources that facilitated the conduct of the present study.

I express my humble gratitude to **Dr. (Mrs). Gowri Ramakrishnan M.Sc., M.Phil., Ph.D., Registrar,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing all facilities necessary for the study.

I am also thankful to **Dr. (Mrs). R. Parvatham M.Sc., Dip.Ed. M.Phil., Ph.D., Dean Faculty of Science,** for granting the facility required.

I wish to place on record my deep sense of gratitude to **Dr.(Mrs).G.Padmavathi M.Sc., M.Phil., Ph.D.,** Professor and Head, Department of Computer Science, for providing all the facilities to complete the project.

I owe great deal of gratitude to my esteemed guide **Mrs.N.Valliammal M.Sc., M.Phil., Assistant Professor ,** Department of Computer Science, for imparting the tremendous assistance and well timed support for triumph of my project.

I take this unique opportunity to express my sincere thanks to my project Coordinator **Dr.(Mrs).V.Srividhya M.Sc., M.Phil., Ph.D., Assistant Professor,** Department of Computer Science, for her kind advice and knowledgeable suggestion, which helped me to complete my project successfully.

I express my gratitude to **Dr.(Mrs).Saroja Prabakaran, M.A, Dip.Ed.,Ph.D., Director, Hall of Residence,** for her support in completion of project.

I would extend my hearty thanks to one and all who helped me directly or indirectly for successful completion of my project.

SYNOPSIS

SYNOPSIS

The goal of partial duplicate image discovery is to find groups of images in a large dataset that contain the same object, which may not necessarily occupy the entire image. Our ultimate goal is to detect partial-duplicate images from a web scale image database, where both precision/recall and computational cost are critical for scalability. This paper, aimed to exploit locality and geometric constraints to improve precision/ recall and to reduce computational cost. In existing system they proposed Partition min-Hash (PmH), a novel hashing scheme to exploit locality, i.e. the fact that duplicate regions are usually localized in an image. In PmH, an image is first divided into overlapping partitions. Hashing is then applied independently to the visual words within each partition to compute a min-hash value. In proposed work to improve the efficiency a novel hyper sphere based hashing function, spherical hashing, to map more spatially coherent data points into a binary code compared to hyper plane-based hashing functions. Furthermore, in proposed a new binary code distance function, spherical Hamming distance, that is tailored to the hyper sphere based binary coding scheme, and design an efficient iterative optimization process to achieve balanced partitioning of data points for each hash function and independence between hashing functions.

CONTENTS

CONTENTS

INTRODUCTION

1. INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

The definition of a near duplicate image varies depending on what geometric variations are deemed acceptable. The application ranges from exact duplicate detection where no changes are allowed to a more general definition that requires the images to be of the same scene, but with possibly different viewpoints and illumination. The project builds on a min-Hash method that addresses (through a similarity threshold parameter) a whole range of near duplicate images: from images that appear, to a human observer, to be identical or very similar to images of the same scene or object. Detection of near duplicate images in large databases imposes two challenging constraints on the methods used. Firstly, for each image only a small amount of data (a fingerprint) can be stored; secondly, queries must be very cheap to evaluate. Ideally, enumerating all the duplicates of an image should have complexity close to linear in the number of duplicates returned.

The choice of an image representation and a distance measure affects both the amount of data stored per image and the time complexity of the database search. The amount of stored data ranges from a constant (small) amount of data per image to storing large sets of image features, whose size often far exceeds the size of the images themselves. When searching the database for relevant images, algorithms of different time complexity are used, the most naive approach being computing the similarity between every image pair in the database. Recently, a bag of visual words with tf-idf (term frequency – inverse document frequency) weighting has proven to be a very successful approach for image and particular object retrieval, even on large database. The tf

part of the weighting scheme captures the number of features described by a given visual word. The frequency of visual word in the image provides useful information about repeated structures and textures. The idf (image difference feature) part captures the in formativeness of visual words – visual words that appear in many different images are less informative than those that appear rarely.

For instance, a duplicate image can be defined as the exact syntactic terms and sequence, with or without formatting differences. Thus, there are either no-differences, i.e. identical, or only formatting differences, and the content of the data is exactly the same. In such a restrictive definition of duplicates, can calculate a unique hash value or each data item and then examine for duplications by looking up the hash value (checksum) in either an in-memory hash or a persistent lookup system. Specific to images/videos, a duplicate is often not an identical copy but rather a transformed copy of the original source of images/videos using digital geometric transformations. In the literatures, the transformed copies are sometimes referred to as near duplicates of the source images/videos. While some of the transformations are quite exotic – to ensure that each variant is unique and thus would not be detected by existing algorithms, but visually all variants are very similar to the source image and to each other.

The project reveals in near duplicate image detection (NDID), specifically in the min-Hash algorithm. The algorithm originates from text retrieval and was previously used for NDID. The min-Hash method stores only a small constant amount of data per image, and a complexity for duplicate enumeration that is close to linear in the number of duplicates returned. The method represents the image by a sparse set of visual words. Similarity is measured by the set overlap (the ratio of sizes between the intersection and the union). The advantage of such a choice of image representation and the similarity measure is that it enables very efficient retrieval. The

drawback is that some relevant information is not preserved in the set of visual words (binary) representation.

1.2 PROBLEM DEFINITION

Detection of near-duplicate images has been investigated for years. In existing they proposed near-duplicate detection by using the Stochastic Attribute Relational Graphing technique. With an image to image comparison scheme, they show good results in terms of robustness and discrimination. However, the high computational cost limits its application to large-scale databases. The traditional Content-Based-Image-Retrieval (CBIR) framework used for duplicate detection in a large data collection. Although the system is able to handle large-scale database, the proposed global image representation, e.g. color/texture feature, is not fully optimized on near-duplicate detection problem. Recent surveys conducted an extensive comparison between global feature and local feature in near duplicate detection application. Both works conclude that local image descriptors are in general superior to global image descriptors in terms of accuracy, while at the cost of higher complexity of computing the features and more space for storing them.

LITERATURE STUDY

2. LITERATURE STUDY

1. Near duplicate image detection: min-hash and tf-idf Weighting by Chum, O.Philbin, J.Zisserman .A

Introduction:

Two documents are near duplicate if the similarity sim_s is higher than a given threshold r. The goal is to retrieve all documents in the database that are similar to a query document. This section reviews an efficient randomized hashing based procedure that retrieves near duplicate documents in time proportional to the number of near duplicate documents. The outline of the algorithm is as follows: First a list of min-Hashes is extracted from each document. A min-Hash is a single number having the property that two sets A1 and A2 have the same value of min-Hash with probability equal to their similarity sims(A1,A2). For efficient retrieval the min-Hashes are grouped into n-tuples called sketches. Identical sketches are then efficiently found using a hash table. Documents with at least h identical sketches (sketch hits) are considered as possible near duplicate candidates and their similarity is then estimated using all available min-Hashes.

The min-Hash method stores only a small constant amount of data per image, and a complexity for duplicate enumeration that is close to linear in the number of duplicates returned. The method represents the image by a sparse set of visual words. Similarity is measured by the set overlap (the ratio of sizes between the intersection and the union). The advantage of such a choice of image representation and the similarity measure is that it enables very efficient

retrieval. The drawback is that some relevant information is not preserved in the set of visual words (binary) representation

The definition of a near duplicate image varies depending on what photometric and geometric variations are deemed acceptable. The application ranges from exact duplicate detection where no changes are allowed to a more general definition that requires the images to be of the same scene, but with possibly different viewpoints and illumination. In this research, we build on a min-Hash method that addresses (through a similarity threshold parameter) a whole range of near duplicate images: from images that appear, to a human observer, to be identical or very similar to images of the same scene or object. Detection of near duplicate images in large databases imposes two challenging constraints on the methods used. Firstly, for each image only a small amount of data (a fingerprint) can be stored. Secondly, queries must be very cheap to evaluate. Ideally, enumerating all the duplicates of an image should have complexity close to linear in the number of duplicates returned.

The similarity function is a set overlap that is weighted by the word weights, i.e. words with low weight (common to most of the documents) contribute to the similarity less than rare, discriminative words. As a second step towards tf-idf weighting, we propose extending the similarity measure to compute a weighted histogram intersection score, which is able to take the term frequency into account.

Advantages:

- Image representation and the similarity measure is that it enables very efficient retrieval.
- Similarity measures do not require extra computational effort compared to the original measure.

Disadvantages:

- The drawback is that some relevant information is not preserved in the set of visual words (binary) representation.
- Firstly, for each image only a small amount of data (a fingerprint) can be stored.
- Secondly, queries must be very cheap to evaluate. Ideally, enumerating all the duplicates of an image should have complexity close to linear in the number of duplicates returned.

2. Geometric min-hashing: Finding a needle in a haystack by Chum,

O. Perdoch, M.Matas .J

Introduction:

Algorithms based on hashing techniques are the core of methods that have produced impressive results for a range of computer vision problems, like matching of point sets , object recognition , image retrieval , duplicate detection and clustering in large image collections . In the paper, we propose a novel hashing scheme – the Geometric min-Hash (GmH). The advantages of the Geometric min-Hashing have high impact in problems involving significant occlusion or small physical overlap of the viewing fields. In such cases the difference in recall and precision reach the magnitude and compared to min-Hash algorithm. Moreover, GmH is less sensitive to scale changes. The advantages of the min-Hash, e.g. compact representation and robustness, are preserved. The potential of the method is demonstrated on small object discovery in a large unordered collection of images.

The min-Hash describes images by selecting independently visual words as global descriptors, with the property that the higher number of common features in two images, the higher the probability of having the same min-Hash. A single min-Hash is not sufficiently discriminative to support indexing and thus min-Hashes must be grouped into stuples (sketches) for hashing. In order to find a pair of possibly related images (objects, windows), all s min-Hashes in a sketch must agree. Increasing s reduces both the number of random collisions and true hits exponentially. Due to different "half-times" in the exponential reduction of collisions, the ratio of true to false positives increases rapidly too. On the other hand, the "and" operation exponentially reduces the probability of retrieving all instances of the image or object.

Advantages:

- Geometric min-Hash is compared to min-Hash by achieving higher precision and recall.
- Selected feature provides information to select a second min-Hash with much higher repeatability.

Disadvantages:

- The representation of the geometric hash is not very compact and it is difficult to scale to hundreds or thousands of images.
- Min-Hash in the s-tuple is not repeated.

3. Efficient near-duplicate detection and sub-image retrieval by Ke,

Y. Sukthankar, R.Huston.L

Introduction:

Image manipulation software becomes more powerful and user-friendly; pirating images is becoming increasingly easy. Although digital watermarking techniques exist, these schemes are very difficult to design and there is an inherent trade-off between the robustness of the watermark and the amount of degradation induced in the image. To circumvent digital watermarking, the pirated images are often altered slightly — for instance, by cropping and rescaling. The problem of matching a slightly altered photograph to its original is termed near-duplicate image detection. A photo publishing agency could use a system automatically to identify the potential copyright violations and dispense with digital watermarks altogether. A more insidious form of image manipulation is becoming increasingly popular in propaganda, is the creation of fake photographs by cutting and pasting pieces extracted from different original sources. For instance, one could crop political figures from two different photographs and create a fake composite image showing them shaking hands, even though the individuals may never have met in reality. The problem of matching a small portion of one image to its original is termed sub-image retrieval. If the original images were stored in our system, we could detect query images that were composites and accurately identify the exact sources used in their creation.

We believe that practical systems that address the applications discussed above should satisfy the following requirements:

1. High recall. All images in the database that contain sub images that are present in the query image should be found, even if the sub-images only occupy a small portion of the original.

2. High precision. If the database images and the query image do not have sub-images in common, then they should not be matched. This is important because incorrectly-flagged images will waste the user's time.

3. Efficiency. The time needed to query an image should be small, enabling the system to scale to very large databases.

The research that is most similar to the local features are extracted from images and matched using an approximate similarity search. Our system differs significantly in the following respects. First, scale and rotation-invariant interest point detectors are used and more distinctive local descriptors, and perform geometric verification on the matched features. Second, instead of an ad hoc approximate similarity search, we employ locality-sensitive hashing, an algorithm with provable performance bounds. These contribute to the dramatic improvements in accuracy. Third, we build offline indices that are optimized for disk access and search for all of the query local descriptors in a single pass. This enables us to query large image collections in interactive time.

The DoG (derivative of Gaussian) detector consists of three major stages: (1) scale-space peak selection; (2) interest point localization; (3) orientation assignment. In the first stage, potential interest points are identified by scanning the image over location and scale. This is implemented efficiently by constructing a Gaussian pyramid and searching for local peaks, termed key points, in a series of difference of Gaussian images. In the second stage, candidate key points are localized to sub-pixel and sub scale accuracy, and eliminated if found to be

unstable. The third stage identifies the dominant orientations for each key point based on its local image patch. The assigned orientation(s) scale and location for each key point enables us to construct a canonical view for the key point that is invariant to similarity transforms.

The use of local descriptors has several characteristics that are ideal for solving the near-duplicate image detection problem. First, the interest points are scale and rotation invariant. This allows detecting and matching the same set of interest points even after images have been arbitrarily rotated or scaled. Our approach is also robust to deformations such as Gaussian blurring, median filtering, and the addition or removal of noise, which can degrade or destroy the high frequency content of the original image. This is because a subset of interest points in the original image will continue to match those interest points that encode lower frequency content in the transformed image (corresponding to larger image areas). Second, the descriptors are robust to image deformations such as affine warp, changes in brightness and contrast, etc. Furthermore, PCASIFT ignores color and operates on gray-scale images, making the algorithm immune to transforms that manipulate the color content of the image, such as saturation and colorization. Finally, because we use local descriptors, our system can find matches even if there is significant occlusion or cropping in the images. The system requires as few as five interest points (out of hundreds) to match between two images in terms of descriptor similarity and geometric constraints. Despite the small number of interest points needed to match, we maintain a low false positive rate because the local descriptors are highly distinctive and the geometric constraints further discard many false positives.

Advantages:

- Approximate similarity search that is well suited for high dimensional data.
- Traditional data structures for similarity search suffer from the curse of dimensionality.

Disadvantages:

- The problem of matching a small portion of one image to its original is termed sub-image retrieval.
- If the database images and the query image do not have sub-images in common, then they should not be matched.

SYSTEM REQUIREMENTS

3. SYSTEM REQUIREMENTS

The section describes the hardware and software specification needed for both development and implementation phases of this project.

3.1 HARDWARE SPECIFICATION

PROCESSOR	:	INTEL PENTIUM IV
RAM	:	512 MB
HARD DISK	:	80 GB HDD
DISPLAY TYPE	:	SVGA
KEYBOARD	:	110 KEYS/ (LOGITECH)
MOUSE	:	LOGITECH MOUSE/OPTICAL
MONITOR	:	SONY

3.2 SOFTWARE SPECIFICATION

PLATFORM	:	WINDOWS 7
APPLICATION DEVELOPMENT	:	MATLAB

3.3 OVERVIEW OF THE SOFTWARE

MATLAB (Matrix Laboratory) is a numerical computing environment and fourth-generation programming language. Developed by Math Works, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

MATLAB provides an intuitive language and a flexible environment for technical computations which integrates mathematical computing and visualization tools for data analysis and development of algorithms and applications. This program system currently features more than 600 mathematical, statistical, and engineering functions. Its open architecture and companion products allow users to explore, develop, share and modify data sets, algorithms and custom tools, thereby achieving a fairly fast updating, improving and expanding of the MATLAB environment.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

In 2004, MATLAB had around one million users across industry and academia. MATLAB users come from various backgrounds of engineering, science, and economics. MATLAB is widely used in academic and research institutions as well as industrial enterprises.

MATLAB was first adopted by researchers and practitioners in control engineering, Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra and numerical analysis, and is popular amongst scientists involved in image processing.Matlab is used only for mathematical analysis in image processing and not for implementation of image processing software.Moreover,Matlab is used to solve some numeric problems but should not be used to simulate computer architectures, systems software and computer networks.

Syntax:

The MATLAB application is built around the MATLAB language. The simplest way to execute MATLAB code is to type it in the Command Window, which is one of the elements of the MATLAB Desktop. When code is entered in the Command Window, MATLAB can be used as an interactive mathematical shell. Sequences of commands can be saved in a text file, typically using the MATLAB Editor, as a script or encapsulated into a function, extending the commands available.

Structures:

MATLAB supports structure data types. Since all variables in MATLAB are arrays, a more adequate name is "structure array", where each element of the array has the same field names. In addition, MATLAB supports dynamic field names (field look-ups by name, field manipulations etc). Unfortunately, the MATLAB JIT compiler is powerful, but not very smart, it can only compile very simple operations so,MATLAB JIT does not support MATLAB

structures; therefore just a simple bundling of various variables into a structure will come at a cost.

FEATURES

- Simple programming environment including control structures like loops and selections.
- Linear algebraic system solvers for vectors, matrices and functionality like computing products and inverses, scaling, summing, multiplications, factorizations, and so on.
- Simple graphic interface for visualization and simulation. Open component system based on toolboxes which can be created, modified, customized and shared by the users.
- High-level language for technical computing Development environment for managing code, files, and data Interactive tools for iterative exploration, design, and problem solving Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration 2-D and 3-D graphics functions for visualizing data Tools for building custom graphical user interfaces .Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft Excel Development Environment(5:02)
- Ability to zip and unzip files and folders in the Current Folder browser to simplify sharing of files.
- New visual cues in the Current Folder browser to show directories on the MATLAB path.
- Enhanced tab completion in the MATLAB Editor with support for local variables, sub functions, and nested functions.

- Expanded access in the plot selector to plots from the Curve Fitting, Filter Design, Image Processing, and Signal Processing Toolboxes.
- Enhanced File and Folder Comparison Tool, highlighting changes within lines in file comparisons, and sorting results by name, type, size, or timestamp in folder comparisons Performance and Large Data Set Handling.

- Multithreaded computation support for long vector fft, conv2, integer conversion, and integer arithmetic functions.

Performance improvements for mrdivide, convn, histc, sortrows, and sparse matrix indexed assignment

ADVANTAGES OF MAT LAB

- Entrance time to start using MATLAB is low. After only a few hours of training, a new MATLAB user can start developing simulation tools.

- Recent versions of the MATLAB compiler can compile to C, C++ and binary code, allowing the use of different optimization options for high-speed executables.

- As a consequence, performance of the MATLAB linear-algebraic solvers has been drastically improved in the latest versions of the software, which include also additional accelerators for operations with vector/matrix data types.

- The open architecture allows for very rapid extension of the range of functionality of MATLAB by developing and sharing new toolboxes.

- Using MATLAB tools and toolboxes, it is possible to develop a prototype of an application for a relatively very short time.

- It is easy to exchange toolboxes and parts of code within a team and between teams working in the same area.

DISADVANTAGES OF MATLAB

- Mat lab is an interpreted language.

- The main disadvantage of interpreted languages is execution speed. When a language is compiled, all of the code is analyzed and processed efficiently, before the programmer distributes the application. With an interpreted language, the computer running the program has to analyze and interpret the code (through the interpreter) before it can be executed (each and every time), resulting in slower processing performance.

SYSTEM STUDY

4. SYSTEM STUDY

4.1 EXISTING SYSTEM

In this system, based on the observation that duplicate regions among partial-duplicate images are usually localized, we propose a new method called Partition Min-Hash. It has better precision and recall and runs orders of magnitude faster than standard min-hash. It is also very easy to implement and only has partition size and overlap as tuning parameters. The rest of this section introduces the method and discusses its performance. An image is divided into p rectangular regions of equal area, which we call partitions. Then, instead of extracting min-hash sketches from the entire image, min one extracted from each partition, are inserted into the hash table. As will be analyzed in the following section, partitions that fully and tightly capture a duplicate region between images lead to better precision and recall, compared to cases in which a duplicate region spans several partitions, or where the partitions are much larger than the duplicate region. With evenly divided partitions, the duplicate is often split into two or more partitions. To alleviate this, the design partitions are overlapping and multi-scale. This gives us a better likelihood of having at least one partition that captures the duplicate region completely. This may remind the reader of the sliding window technique, widely used for object detection. The spirit is, in fact, similar to that are hoping for one of the sub windows to hit a region of interest, which, in our case, is an unknown duplicate region. Avoidance of redundant computation on overlapping partitions by precomputing and reusing min-hashes. An image is divided into a grid, where the grid elements are the greatest common regions among partitions that cover that region.

Disadvantages:

- Have to improve the nearest search method.

4.2 PROPOSED SYSTEM

Many binary code encoding schemes based on hashing have been actively studied recently, since it can provide an efficient similarity search, especially nearest neighbor search, and compact data representations suitable for handling large scale image databases in many computer vision problems. Existing hashing techniques encode high dimensional data points by using hyperplane-based hashing functions. The project propose a novel hypersphere based hashing function, spherical Locality Sensitive hashing, to map more spatially coherent data points into a binary code compared to hyperplane-based hashing functions. Furthermore, in propose a new binary code distance function, spherical Hamming distance, that is tailored to our hypersphere based binary coding scheme, and design an efficient iterative optimization process to achieve balanced partitioning of data points for each hash function and independence between hashing functions. The extensive experiments show that our spherical hashing technique significantly outperforms six state-of-the-art hashing techniques based on hyperplanes across various image benchmarks of sizes ranging from one to 75 million of GIST descriptors. The performance gains are consistent and large, up to 100% improvements. The excellent results confirm the unique merits of the proposed idea in using hyperspheres to encode proximity regions in high-dimensional spaces. Finally, this method is intuitive and easy to implement.

Advantages:

- It does not need dimensionality reduction.
- Spherical Locality Sensitive Hashing is the best method for nearest neighbor search in large database compared to the existing system.

SYSTEM DESIGN

5. SYSYTEM DESIGN

System design is the process of planning a new system to complement or altogether replace the old system. The purpose of the design phase is the first step in moving from the problem domain to the solution domain. The design of the system is the critical aspect that affects the quality of the software. The design phase translates the logical aspects of the system into physical aspects of the system.

5.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

5.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner and the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

5.3 DATASET

The dataset contains the images like flowers, butterfly, horse, bus, dinosaurs. This work contains the input and the output images only for horses. The output images will be the related images of horses. Here, the input images are collected from the web and manually categorized them into some categories, where the images within each category are partial duplicates. There are no exact duplicates among this set of that images.

SYSTEM DEVELOPMENT

6. SYSTEM DEVELOPMENT

6.1 METHODOLOGY

6.1.1 Partition Min-Hash:

Based on the observation that duplicate regions among partial-duplicate images are usually localized, we propose a new method called Partition Min-Hash. It has better precision and recall and runs orders of magnitude faster than standard min-hash. It is also very easy to implement and only has partition size and overlap as tuning parameters. The rest of this section introduces the method and discusses its performance.

Method Details:

An image is divided into p rectangular regions of equal area, which is called as partitions. Then, instead of extracting min-hash sketches from the entire image, min hash sketches are extracted for each partition independently. The p min-hash sketches, one extracted from each partition, are inserted into the hash table. As will be analyzed in the following section, partitions that fully and tightly capture a duplicate region between images lead to better precision and recall, compared to cases in which a duplicate region spans several partitions, or where the partitions are much larger than the duplicate region. With evenly divided partitions, the duplicate is often split into two or more partitions. To alleviate this, we design partitions to be overlapping. This gives us a better likelihood of having at least one partition that captures the duplicate region completely. This may remind the reader of the sliding window technique, widely used for object detection. The spirit is, in fact, similar: we are hoping for one of the sub windows to hit a region of interest, which, in our case, is an unknown duplicate region.

It avoids redundant computation on overlapping partitions by precomputing and reusing min-hashes. An image is divided into grids, where the grid elements are the greatest common regions among partitions that cover that region. Min-hash sketches are precomputed for each grid. Then the min-hash sketch for a partition P is computed by looking up grids fwig that are associated with that partition P and picking the true min-hash sketch among the precomputed min-hash sketches on grids:

$$h(P) = \min_i \{h(w_i) | w_i \in P\} \qquad \text{Equ} \longrightarrow 1$$

The entire algorithm is summarized in Algorithm:

Initialize N_s independent min-hash sketch functions h_i, where $i = 1, \cdots, N_s$.
Initialize N_s hash tables T_i, which map a min-hash sketch s to image ID k.
for all Images I_k in database **do**
  Divide image I_k into a grid, $I_{k,j}$, where $j = 1, \cdots, N_g$
  For each partition, determine the grid elements that are associated with the partition.
  for $i = 1, \cdots, N_s$ **do**
    for $j = 1, \cdots, N_g$ **do**
      Extract sketch $s_i \leftarrow h_i(I_{k,j})$
    for all Partitions **do**
      Look up sketches s_j extracted from grids that belong to current partition, and select true min-hash sketch s^*
      Add (s^*, k) to hash table T_i

6.1.2 Geometric Sensitive Hash:

A typical application of min hash is to use the collisions as cluster seeds that will be expanded (by image retrieval) into complete clusters of partial-duplicate images. In doing so, it is important to reduce false positives in these seeds before they are verified by full geometric

verification, especially for large scale data set where the number of false positives tends to increase. The local geometric structure of features in duplicate regions is usually preserved across images. For example, the top of the Eiffel tower is always above the base, and the word “Starbucks” is always above the word “coffee”. Instead of verifying these local geometric relationships after sketch collisions are retrieved, we encode such local geometric structure into the sketches, so that they can be checked at an earlier stage. This reduces the number of false positive collisions, and therefore reduces the number of full geometric verifications that need to be performed after expansion by image retrieval, saving computational expense. We encode geometric structure into the sketches by hashing the geometric relationship between features. This is achieved by creating an integer ID which encodes their relative geometric configuration among visual words in a sketch, and concatenating it to the min-hash sketch. We call this Geometry Sensitive Hashing. There are many ways to hash the local geometric structure using the relative location and scale of features.

It focus on one of the most popular algorithms for performing approximate search in high dimensions based on the concept of locality-sensitive hashing (LSH). The key idea is to hash the points using several hash functions to ensure that for each function the probability of collision is much higher for objects that are close to each other than for those that are far apart. Then, one can determine near neighbors by hashing the query point and retrieving elements stored in buckets containing that point. The LSH algorithm and its variants has been successfully applied to computational problems in a variety of areas, including web clustering, computational biology, computer vision selected articles, computational drug design and computational linguistics. A code implementing a variant of this method is available from the authors. The purpose of this article is twofold. It describes the basic ideas behind the LSH algorithm and its

analysis and also gives an overview of the current library of LSH functions for various distance measures. Then, we describe a recently developed LSH family for the Euclidean distance, which achieves a near-optimal separation between the collision probabilities of close and far points. An interesting feature of this family is that it effectively enables the reduction of the approximate nearest neighbor problem for worst-case data to the exact nearest neighbor problem over random (or pseudorandom) point configuration in low-dimensional spaces.

Currently, the new family is mostly of theoretical interest. This is because the asymptotic improvement in the running time achieved via a better separation of collision probabilities makes a difference only for a relatively large number of input points. Nevertheless, it is quite likely that one can design better pseudorandom point configurations which do not suffer from this problem. Some evidence for this conjecture is presented, where it is shown that point configurations induced by so called *Leech lattice* compare favorably with truly random configurations.

6.1.3 Locality Sensitive Hashing:

Locality Sensitive Hashing (LSH) is a widely-used algorithmic tool which brings the classic technique of hashing to geometric settings. It was introduced for general metric spaces in the seminal work of Indyk and Motwani [IM98]. Indyk and Motwani showed that the important problem of (approximate) nearest neighbor search can be reduced to the problem of devising good LSH families. Subsequently, numerous papers demonstrating the practical utility of solving high-dimensional nearest neighbor search problems via the LSH approach. Several important problems in computational geometry reduce to the approximate near neighbor problem, including approximate versions of nearest neighbor, furthest neighbor, close pair, minimum spanning tree, and facility location. For a short survey of these topics, see Indyk

[Ind04].Regarding the reduction from (near neighbor problem to LSH, it is usual to credit roughly the following theorem to [IM98, GIM99].

6.2 MODULE DESCRIPTION

Several modules were used as follows:

1. Image preprocessing
2. Min-Hash for Partial Duplicate Image Discovery
3. Partition Min-Hash for Partial Duplicate Image Discovery
4. Proposed Spherical Locality Sensitive Hashing
5. Performance evaluation

1. Image preprocessing:

The module, Image pre-processing can significantly increase the reliability of an optical inspection. Several filter operations which intensify or reduce certain image details enable an easier or faster evaluation. Image Processing Toolbox provides reference-standard algorithms for preprocessing and post processing tasks that solve frequent system problems, such as interfering noise, low dynamic range, out-of-focus optics, and the difference in color representation between input and output devices.

2. Min-Hash for Partial Duplicate Image Discovery:

The module presents some background on the min-hash algorithm. Min-hash is a Locality Sensitive Hashing scheme that approximates similarity between sets. When an image is represented as a set of visual words, the similarity between two images can be defined as the Jaccard similarity between the two corresponding sets of visual words I_1 and I_2:

$$sim(I_1, I_2) = \frac{|I_1 \cap I_2|}{|I_1 \cup I_2|}, \quad \text{Equ} \longrightarrow 2$$

The above equation is simply the ratio of the intersection to the union of the two sets. Min-hash is a hash function $h : I \mapsto v$, which maps a set I to some value v. More specifically, a hash function is applied to each visual word in the set I, and the visual word that has minimum hashed value is returned as the min-hash h (I). One way to implement the hash function is by a look-up table, with a random floating-point value assigned for each visual word in the vocabulary, followed by a min operator. The computation of the min-hash of a set I involve computing a hash of every element in the set and the time taken is therefore linear in the size of the set $|I|$. More details on min-hash can be found. Min-hash has the property that the probability of hashing collision of two sets is equal to their Jaccard similarity:

$$P(h(I_1) = h(I_2)) = sim(I_1, I_2). \quad \text{Equ} \longrightarrow 3$$

Since the output of a min-hash function v is actually a visual word, it carries the position and scale information of the underlying local feature for geometric verification.

3. Partition Min-Hash for Partial Duplicate Image Discovery:

The module, based on the observation that duplicate regions among partial-duplicate images are usually localized, we propose a new method called Partition Min-Hash. It has better precision and recall and runs orders of magnitude faster than standard min-hash. It is also very easy to implement and only has partition size and overlap as tuning parameters. The rest of this section introduces the method and discusses its performance. We can avoid redundant computation on overlapping partitions by precomputing and reusing min-hashes. An image is

divided into a grid, where the grid elements are the greatest common regions among partitions that cover that region. Min-hash sketches are precomputed for each grid w_i. Then the min-hash sketch for a partition P is computed by looking up grids $\{w_i\}$ that are associated with that partition P and picking the true min-hash sketch among the precomputed min-hash sketches on grids:

$$h(P) = \min_i \{h(w_i) | w_i \in P\} \quad \text{Equ} \longrightarrow 4$$

4. Proposed Spherical Locality Sensitive Hashing:

In this module, The LSH algorithm relies on the existence of *locality-sensitive hash functions*. Let H be a family of hash functions mapping $\mathbb{R}^d$ to some universe *U*. For any two points *p* and *q*, consider a process in which we choose a function *h* from H uniformly at random, and analyze the probability that $h(p) = h(q)$. The family H is called *locality sensitive* (with proper parameters) if it satisfies the following condition. A family His called (R, cR, $P1$, $P2$)-sensitive if for any two points *p*, $q \subset \mathbb{R}^d$.

$$\text{if } \|p-q\| \leq R \text{ then } \Pr_{\mathcal{H}}[h(q) = h(p)] \geq P_1$$
$$\text{if } \|p-q\| \geq cR \text{ then } \Pr_{\mathcal{H}}[h(q) = h(p)] \leq P_2$$

Equ ⟶ 5

SLSH uses the randomly regular prototype for partitioning the surface of unit hyper sphere. After rotating the prototype at random, then the hash function h(p) is defined as the number assigned to the vertex which is nearest to the p.

5. Performance evaluation:

This module will analyze the speed and performance of partition min-hash and show that it achieves higher precision and recall in less time than standard min-hash. For the sake of comparison, we can keep the number of sketches per image equal for both cases. With the same number of sketches stored in the hash table, the two methods will use the same amount of memory.

SYSTEM TESTING

7. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

7.1 UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

TEST STRATEGY AND APPROACH

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.2 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or one step up software applications at the company level – interact without error.

Test Results:

All the test cases mentioned above are tested successfully. No defects encountered.

7.3 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

CONCLUSION

8. CONCLUSION

In proposed three novel improvements to min-hash for discovering partial duplicate images in a large set of database are partition min-hash, geometric-sensitive hash and the spherical locality sensitive hashing. In this the proposed algorithm is implemented to avoid the nearest neighbor problem when all constraints to lie on the surface of the unit hyper sphere. The algorithm, named SLSH, is based on the LSH scheme, and the output shows the better result than other two algorithms.

SCOPE FOR FUTURE ENHANCEMENT

9. SCOPE FOR FUTURE ENHANCEMENT

- Apart from robustness and accuracy of copy paste image forgery detection, time complexity also plays an important role to evaluate the performance of the system.
- In future work the time complexity can be improved for the block-matching algorithm.
- Hence, a coarse-to-fine approach is applied to propose an enhanced duplicated region detection model by using sequential block clustering.
- Clustering minimizes the search space in block matching.
- This significantly improves time complexity as it eliminates several extra block-comparing operations.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Ailon, N. and Chazelle, “Approximate nearest neighbors and the Fast Johnson-Lindenstrauss Transform”. In Proceedings of the Symposium on Theory of Computing. B. 2006

2. Andoni, A. and Indyk,“Exact Euclidean locality sensitive hashing”. P. 2004.

3. Andoni, A. and Indyk, “Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions”. In Proceedings of the Symposium on Foundations of Computer Science P. 2006..

4. Andoni, A. and Indyk,”Efficient algorithms for substring near neighbor problem”. In Proceedings of the ACM-SIAM Symposium on Discrete Algorithms. 1203–1212. P. 2006.

5. Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, “An optimal algorithm for approximate nearest neighbor searching”. In Proceedings of the ACM-SIAM Symposium on Discrete Algorithms. 573–582. A. 1994.

6. Bentley, “Multidimensional binary search trees used for associative searching”. Comm. ACM 18, 509–517. J. L. 1975.

7. Buhler,” Efficient large-scale sequence comparison by locality- sensitive hashing”. Bioinform. 17, 419–428. J. 2001.

8. Chum, O., Philbin, J., Zisserman, A.”Near duplicate image detection, min-hash and tf-idf Weighting”. In Proceedings of the British Machine Vision Conference. (2008).

9. David C. Lee, Qifa Key, and Michael Isardy Carnegie “Partition min hash for partial duplicate image discovery” Mellon University, Microsoft Research Silicon Valley.

10. Jegou, H., Douze, M., Schmid, C.:” Hamming embedding and weak geometric consistency for large scale image search”. In: ECCV. (2008).

11. K. Terasawa and Y. Tanaka. “Spherical lsh for approximate nearest neighbor search on unit hypersphere”. In Algorithms volume I.

12.Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.:” Object retrieval with large vocabularies and fast spatial matching”. In: CVPR. (2007).

Websites:

- http://www.cs.cmu.edu/~dclee/pub/eccv10lee.pdf
- http://link.springer.com
- http://hal.archives-ouvertes.fr/docs
- http://www.mit.edu/~andoni/LSH
- http://www.cs.utexas.edu/~grauman/courses/spring2008/slides/Marc_Demo.pdf.

Book References:

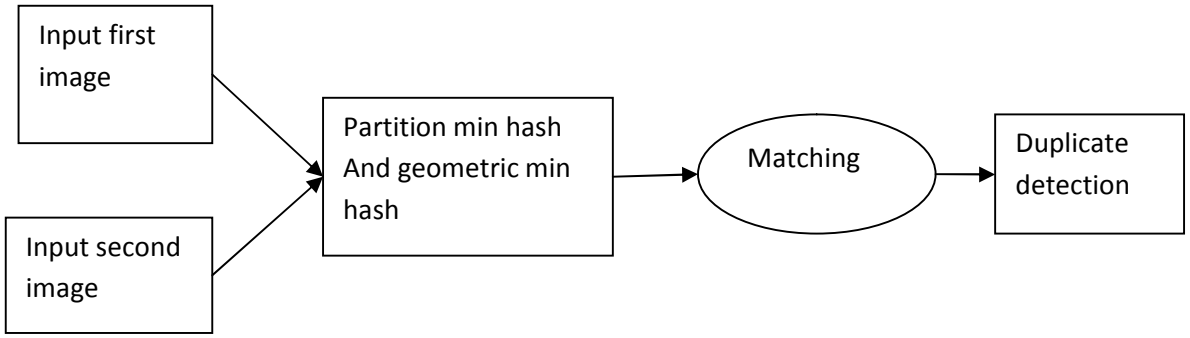
- Digital Image Processing- 3/e (New Edition)- Rafael C. Gonzalez.
- Digital image processing - S. Jayaraman, S. Esakkirajan, T.Veerakumar.

APPENDIX

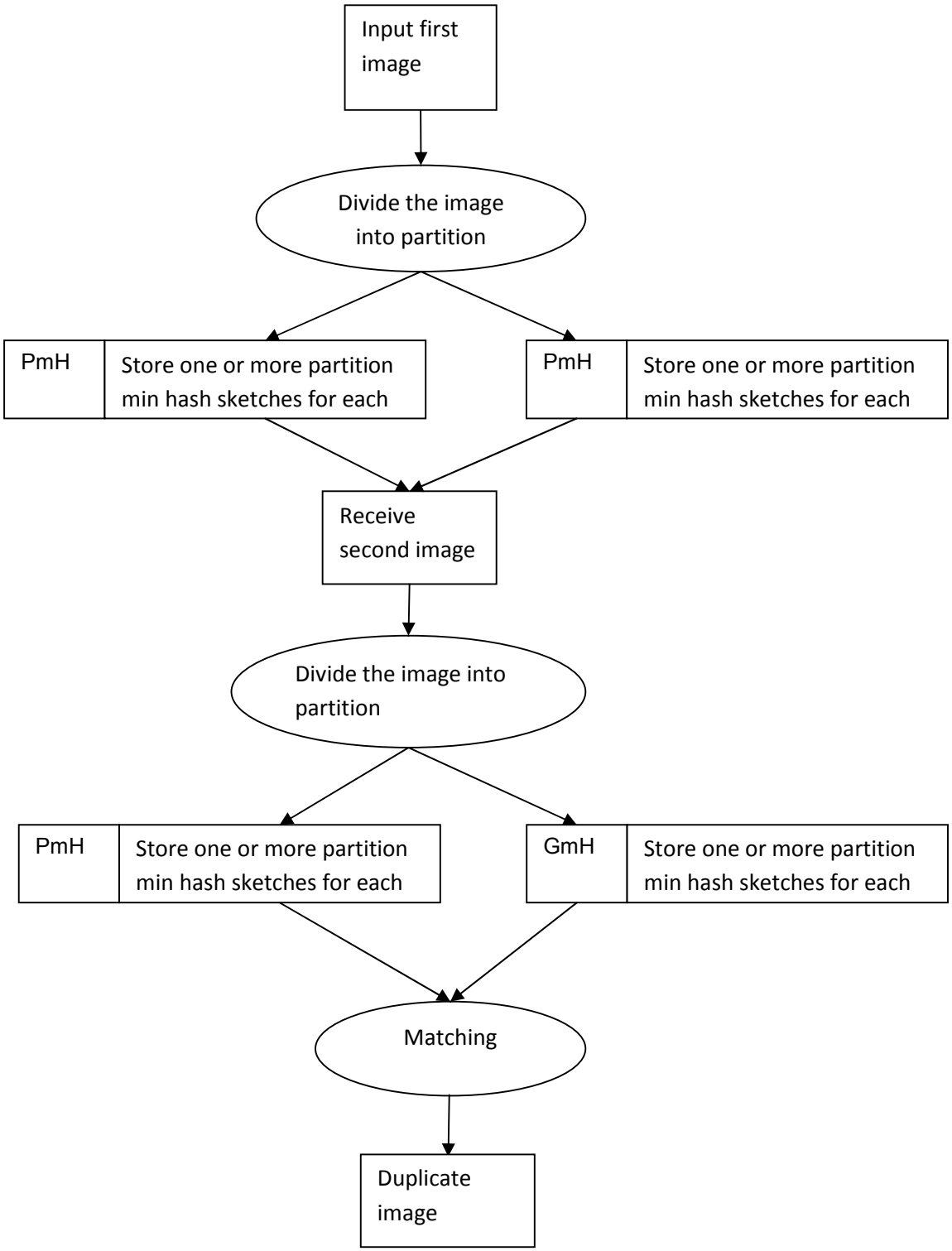
APPENDIX

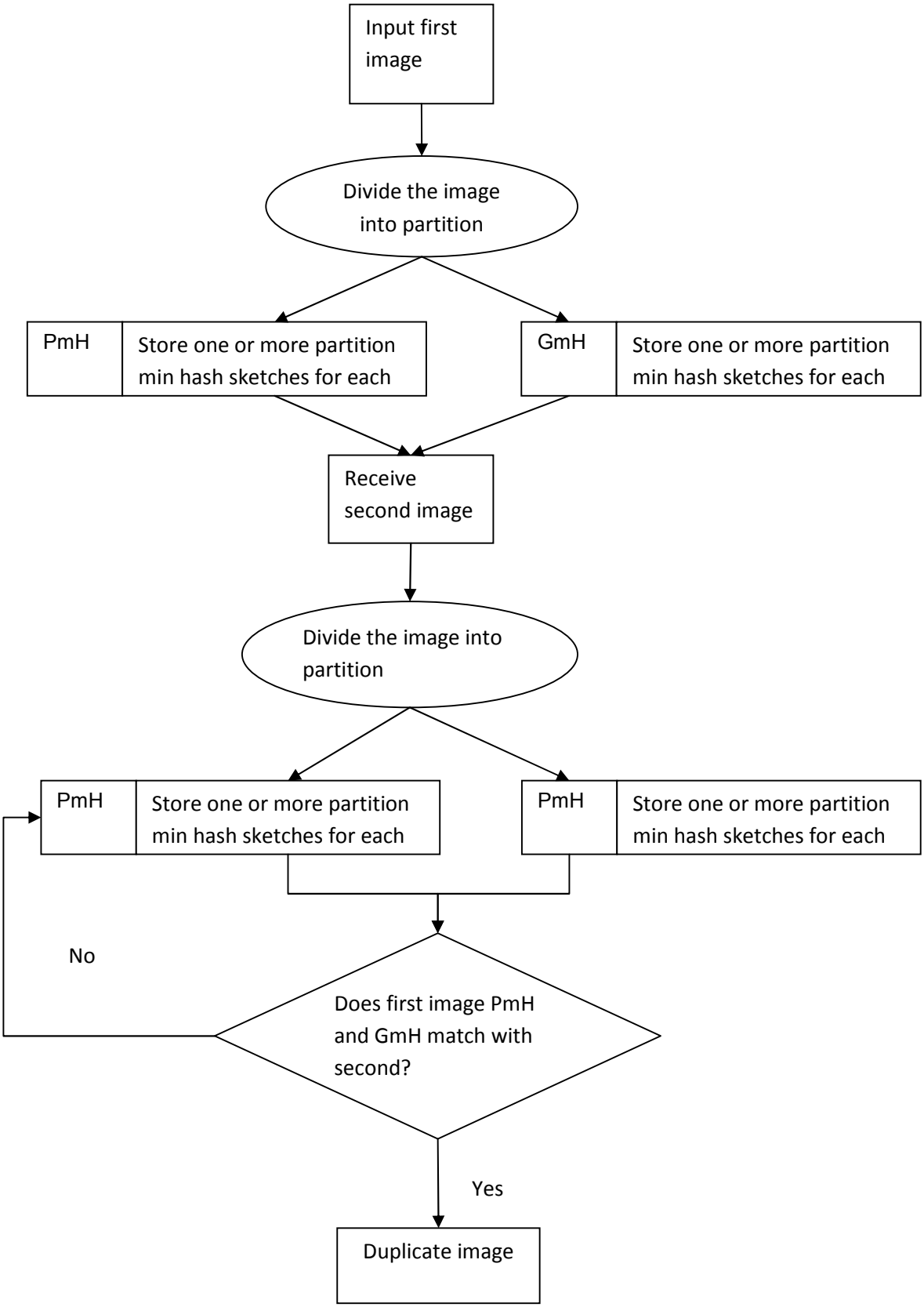
i) DATA FLOW DIAGRAM

Level 0

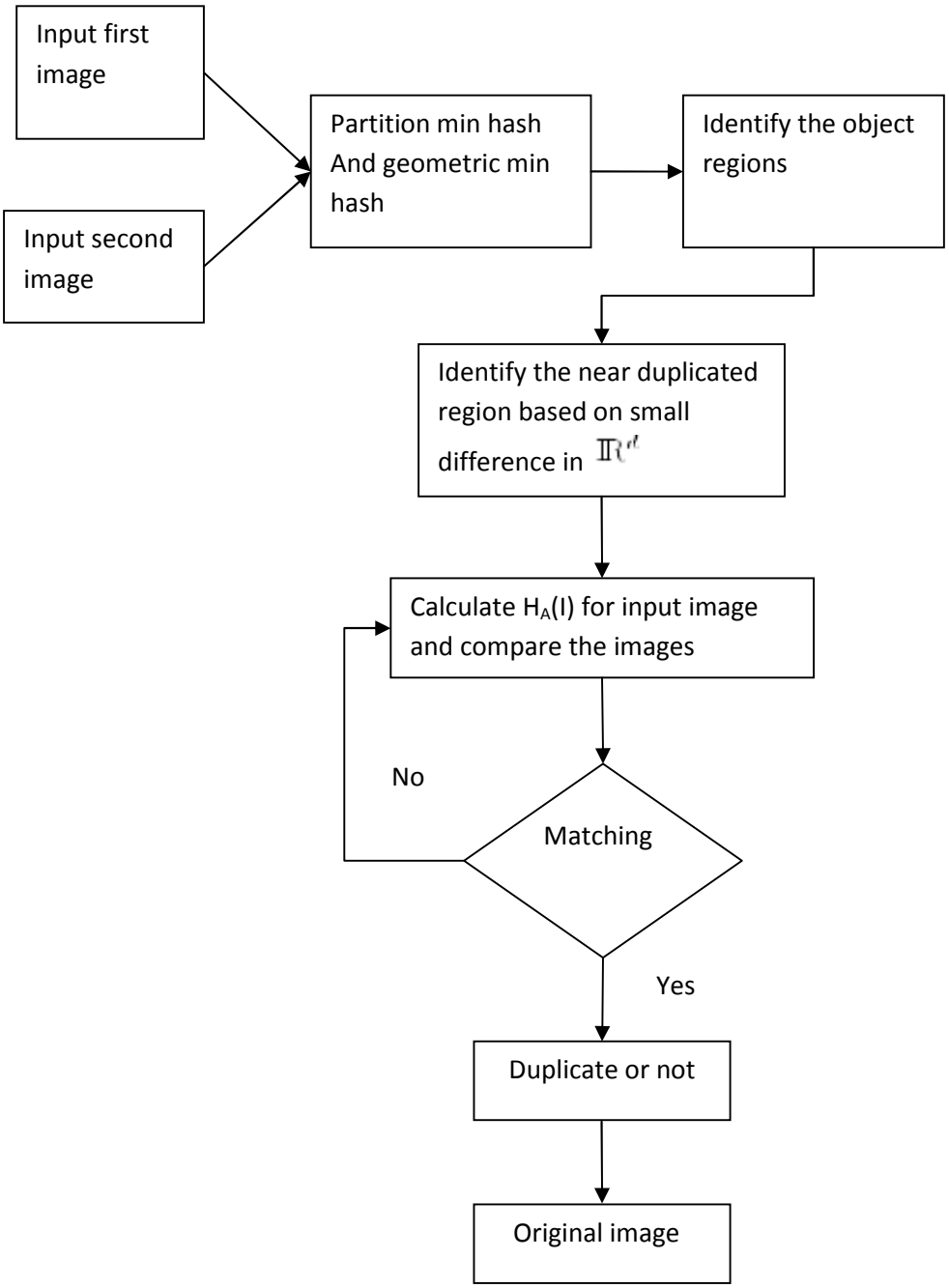


Level 1


Input first image
Divide the image into partition
PmH
Store one or more partition min hash sketches for each
PmH
Store one or more partition min hash sketches for each
Receive second image
Divide the image into partition
PmH
Store one or more partition min hash sketches for each
GmH
Store one or more partition min hash sketches for each
Matching
Duplicate image


Input first image
Divide the image into partition
PmH
Store one or more partition min hash sketches for each
GmH
Store one or more partition min hash sketches for each
Receive second image
Divide the image into partition
PmH
Store one or more partition min hash sketches for each
PmH
Store one or more partition min hash sketches for each
No
Does first image PmH and GmH match with second?
Yes
Duplicate image

Level 3


Input first image
Input second image
Partition min hash And geometric min hash
Identify the object regions
Identify the near duplicated region based on small difference in $\mathbb{R}^d$
Calculate $H_A(I)$ for input image and compare the images
No
Matching
Yes
Duplicate or not
Original image

ii) RESULTS

Input image:

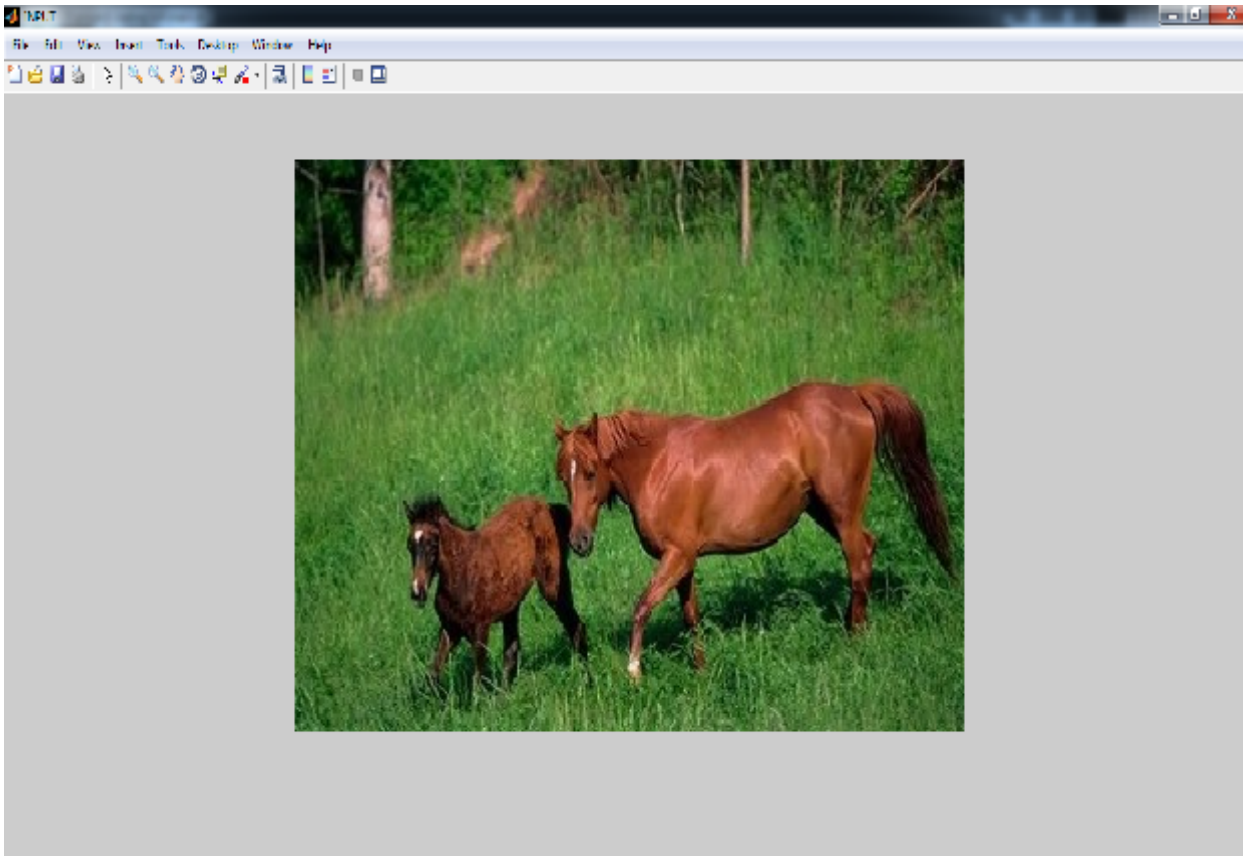


Fig 1 Input image

Preprocessed image:

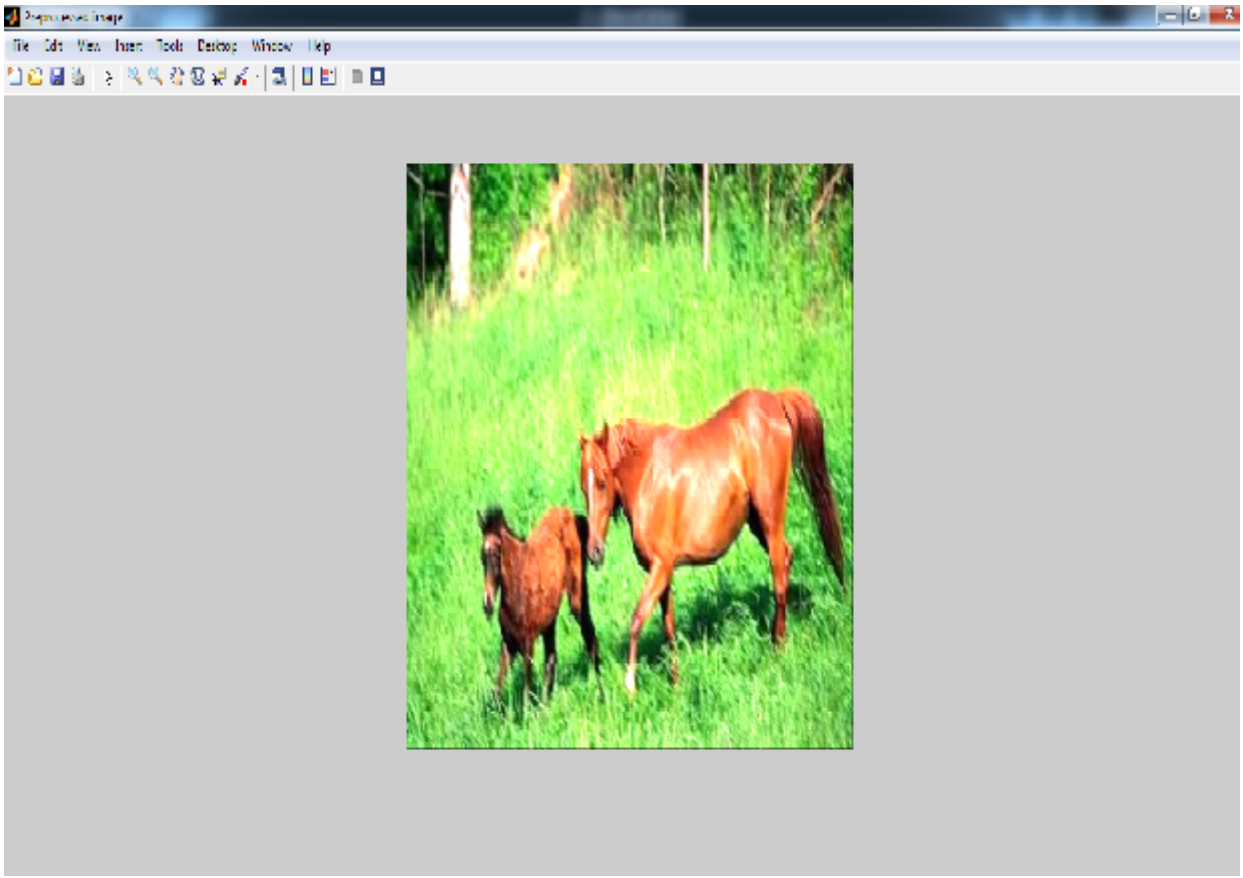


Fig 2 Preprocessed image

Grid image:

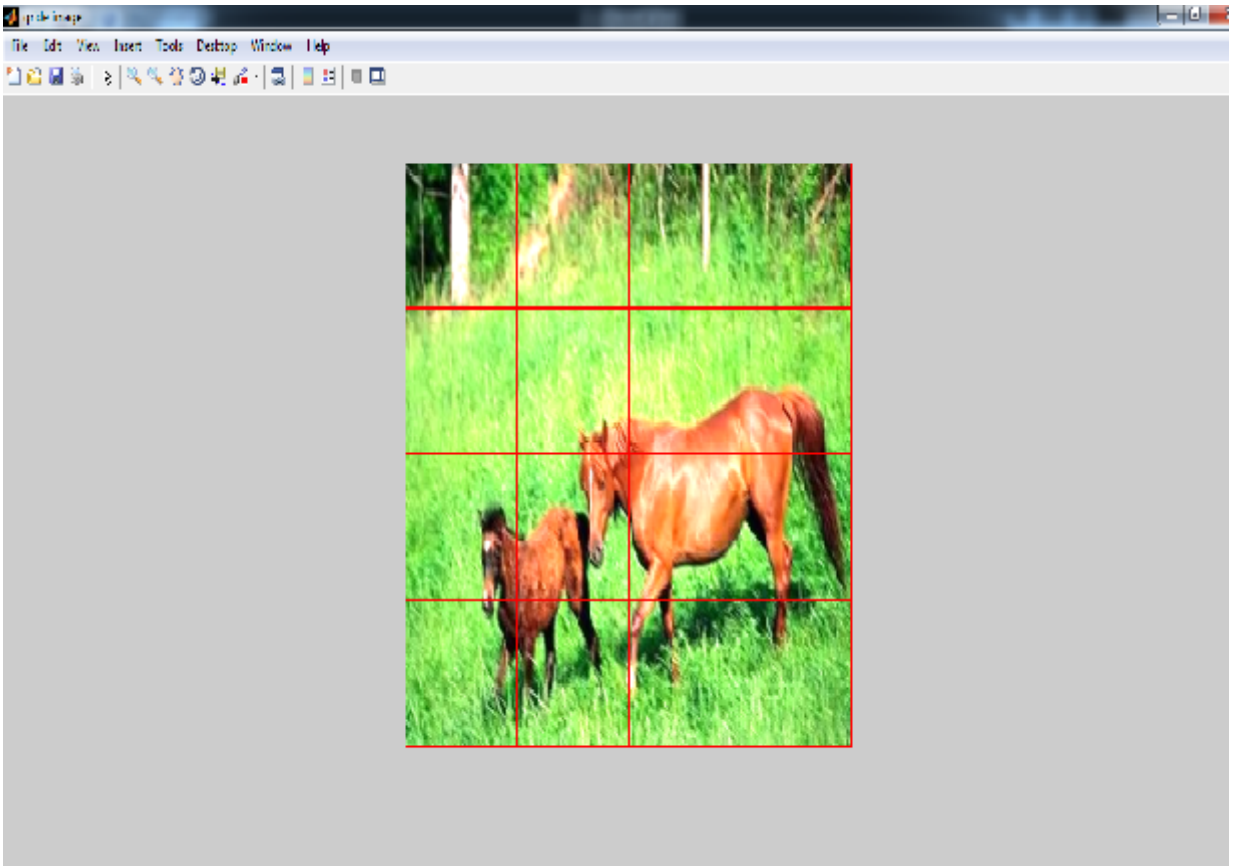


Fig 3 Grid image

Database image:

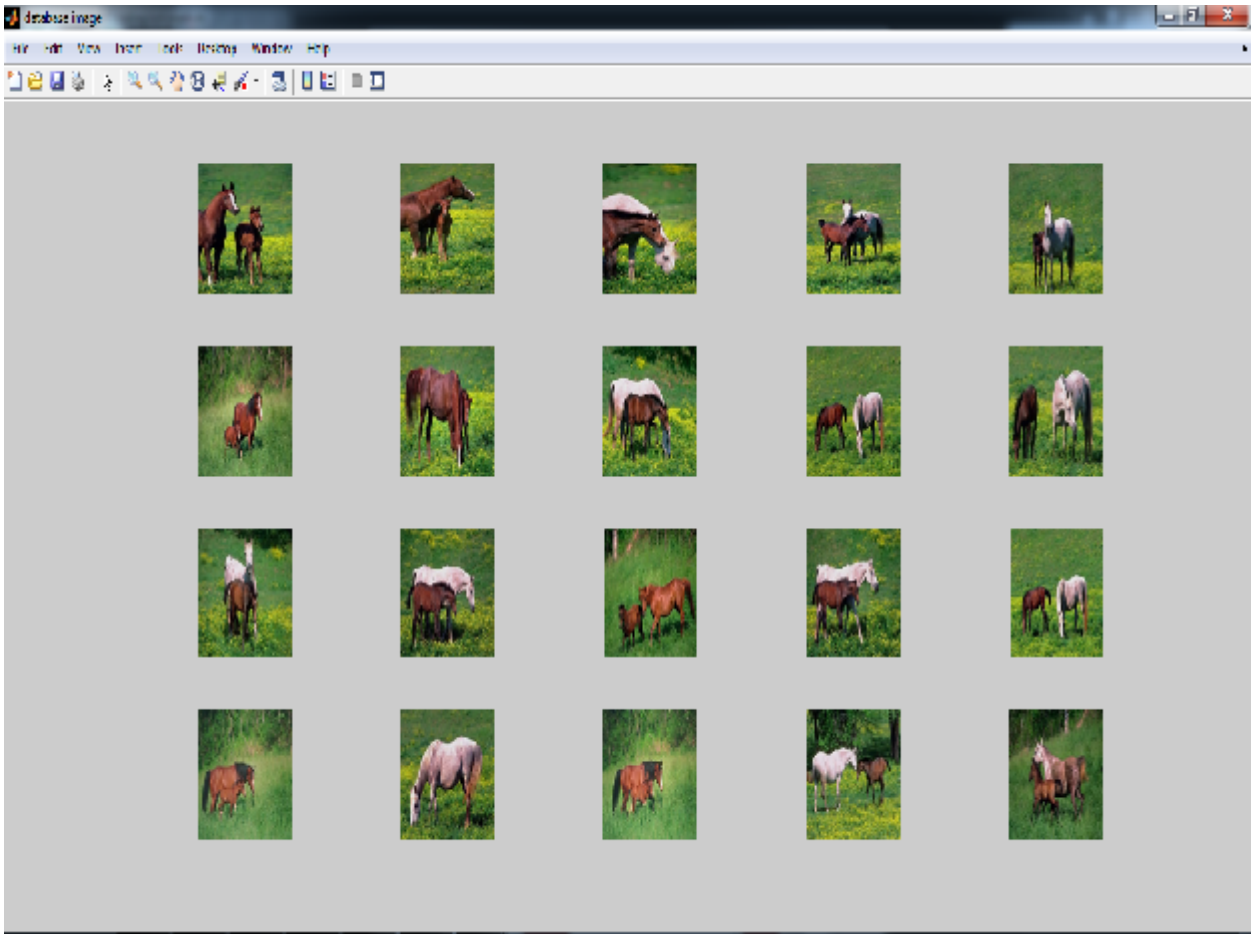


Fig 4 Database image

Accuracy:

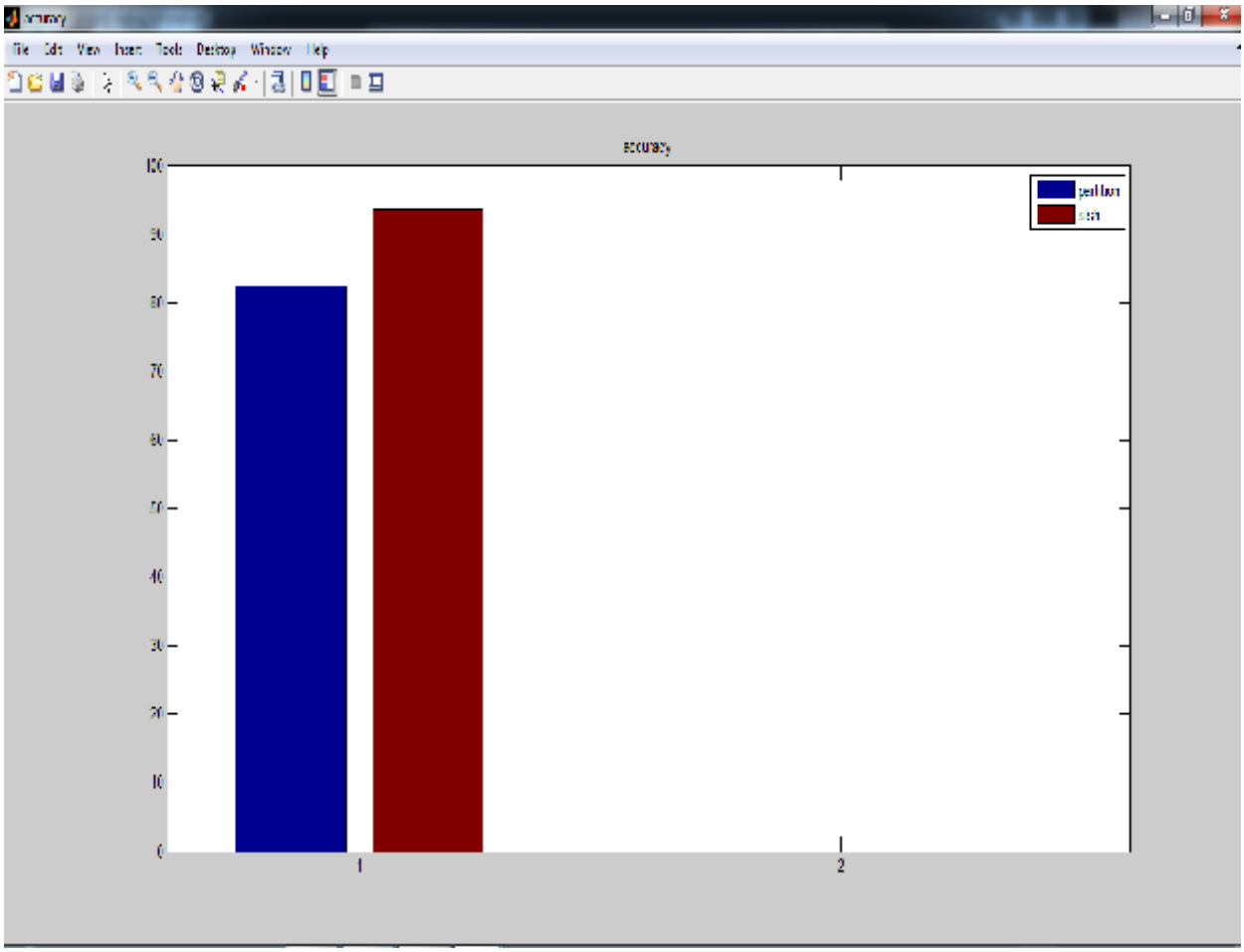


Fig 5 Accuracy