Enabling Eidetic Memory through Image Capturing and Tagging using Google Vision API

BY

NEW JENG MUN

A PROPOSAL

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfilment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEM ENGINEERING (HONS)
INFORMATION SYSTEMS ENGINEERING

Faculty of Information and Communication Technology

(Perak Campus)

JANUARY 2017

DECLARATION OF ORIGINALITY

I declare that this report entitled "**Enabling Eidetic Memory through Image Capturing and Tagging using Google Vision API**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

Acknowledgements

By successfully completing this project, I would like to express my thanks and appreciation to my supervisor, Dr Alex Ooi Boon Yaik. Thanks for all the guidance and suggestion throughout the whole development process.

ABSTRACT

Enabling Eidetic Memory through Image Capturing and Tagging using Google Vision API is a mobile based application system that able to provide user of the similar searching images in local phone storage and having additional tag photo management, which enables user to search, add, delete keyword in photos and retrieve it in a more convenience, shortest duration and more user-friendly way.

In additional, the Image resizer function in the mobile application able to reduce the image size to minimize the best quality of the photos that are needed to send to the cloud to implement the request Google Vision API detection in doing label indexing and abilities to help user to save the maximum data cellular network while uploading photos to perform Google Cloud Vision API services.

The project of this system is trying to overcome the issues that faced by the current system, so that user can reduce time consuming while searching photo in more efficient and effective way with the tag photo management function to help user to organize the label within photos such as to add, to delete and using the image resizer will reduce the maximum data network cellular consumption of the user when sending the request to the Google Cloud Vision API.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

etc	extra
jpg	Joint Photographic Experts Group
HDR	High Dynamic Range

Chapter 1 Introduction

1.1 Motivation

Photographic memory is also called Eidetic Memory which stands for a human that have the ability to recall back all the information or things that she/he seen in an immense detail in their memory. Unfortunately, not many people are born with eidetic memory. With the advancement of smart phones with cameras, people start using the camera to record everyday things from what they eat or lecture slides projected on the screen with the hope to capture information in great detail, accuracy and clarity within the shortest possible time, a single click.

Although such approach is very useful, retrieving the information is not easy, especially those photos are not properly tagged especially over time when there are too many photos taken. As such, the simplest approach to overcome the retrieval issue is to tag all the photos. However, it is not practical for user to tag each photo that they have taken.

Therefore, this project attempts to use Google Vision API to do labelling photos. This project developed a mobile application that able to manage the tags. In addition to that, we also study the impact of photo quality towards Google Vision quality. The purpose of this study is to allow us to reduce the mobile data usage over time while not scarifying the accuracy of Google Vision API. A complete prototype is developed and the effectiveness of such application is put into test.

1.2 Problem statements

Our intention is similar to those photo albums with keyword search function. Existing photo albums search are usually used to search photo online, upload and share their memories and exciting moment to social media. As now with the proliferations of phone with integrated camera and the usage of camera has extended our day-to-day life.

However, the functionalities of existing photo albums search are not designed to be used in the way we wanted:

1. *Search engine functions are usually being used in website to search public photos. But the existing local photo album is not able to perform searchable photo by "text" accurately on what user wants.*

The search filter results in the existing local phone photo album contents are not able to match what user typed as he/she want to perform searchable keywords by text in a more accurate and accessible way because of the imprecise search engine keyword. This is because of the low momentous skilled in search engine development in local phone that did not provide function that able user to create own key tag within the photo to search based on the appropriate keywords they need to retrieve.

2. *The local phone album does not provide any Tag Photo Management label by letting user create, delete own, store it into a database and able to let user to retrieve based on the keyword added by the user.*

On the local photo album, there aren't any features that able user to organize their photos by adding own tag keywords that the user wanted to name it and store it in to database storage and retrieve filter photos based on the keyword added by users or other keywords that which has been predefined by Google Cloud Vision API.

3. *Image processing using Google Cloud Vision API might not be comfortable for many users because of the high consumption of utilization data cellular network while executing the index in each photo.*

High data network cellular consumption is consumed when photo size is larger when user is sending to the Google Cloud Vision API for processing the label indexing keyword in every photo. For instance, one photo may be having few descriptions

to identify the object within it. Without the image sizing method, this will lead to the insufficiencies data network cellular usage.

1.3 Project Objectives

To develop an Eidetic Search mobile application that is able to fulfill the following objectives:

1. User able to search by filtering the existing local phone photo album by using the precise search engine keyword provides by the user. The highly momentous skilled in search engine development in local phone are providing functions that able user to create own key tag in the photo to search based on the appropriate keywords they need to retrieve the photo. The filtering function will provide a more convenient, shortest duration, more user-friendly way and can search photo based on the time filtering such as select the photo capture within 1 day, 1 week, 1 month or etc.

2. Tag photo management in local phone album ably to let user organizing local phone photos by adding own tag keywords that the user wanted to name it, and able to it and update back to database storage. If user's think that the tag is already not necessary anymore, user can delete the tag. The photo can be retrieved based on the keyword added by users or other keywords that which has been predefined by Google Cloud Vision API.

3. Thru using the image resize function, photos in local phone able to reduce the image size to minimize the best quality of the photos that are needed to send and store into the cloud, to implement the request Google Cloud Vision API detection. The notion of this image resizer is to reduce the maximum data network cellular consumption when proceeds to do label indexing in each photo.

1.4 Impact and Contributions

Although there are numerous of mobile based application are doing the same searching images in local phone storage. What really makes this Eidetic Search mobile application unique from others are the objectives, is by enabling eidetic memory through image capturing and tagging using Google Vision API. It's satisfying the user demand by having tag photo management, which enables user to search, add, delete keyword in photos and retrieve it in a more convenience, shortest duration and more user-friendly way. For instance, it will make thorough analyses and filter out the probability score and other detail information within the photo using Google Cloud Vision API Label Detection and categorize it finely by listing the photos from descending order.

By having this improved system, Eidetic Search with significantly improved features such as: By using the image resize function, the mobile application able to reduce the image size to minimize the best quality of the photos that are needed to send to the cloud, to implement the request Google Vision API detection in doing label indexing and abilities to help user to save the maximum data cellular network while uploading photos to perform Google Cloud Vision API services.

Hence, the development of this system is trying to overcome the issues that faced by the current system, so that the user can reduce time consuming, while searching photo in an efficient and effective way, tag photo management to help user to organize the label within photos, like to add, delete and using the image resizer will reduce the maximum data network cellular consumption of the user when sending the request to the Google Cloud Vision API.

Chapter 2 - Literature Review

2.1 Image Search by Keyword

2.1.1 Google Image Search

Google Image Search now has a beneficial feature that able to allow user to search things by images. This is, user can choose any photos from the web or photo of their own collection and trigger searching based on that selective photo to reverse photo searching. Google Googles is the app that allows users to capture photos on their phone and search out for the information needed by the user. It identifies the specific objects within the photo and uses Google Search Engine that act as backend code.

Pros: Allows the user to search images with option such as can choose on images from the user's collection or web.

Cons: User need to click on the "Request Desktop Site" option for the images tap to perform this searching whether in IOS, Android and Windows.

2.1.2 Google Photos

Google Photos is now making our photos favourably searchable and also returns result in a quite accurate based on dates, locations, faces, contents and etc. Google Photo have something comforting to use which are we no need to care about what kind of implementation high-disciplined naming convention or sorting system and ensuring our users are able to retrieve what photo we want by using Google Photos. For instance, if user looking for photos which are taken in Jordan? It is so easy: search "Jordan" and Google Photos will retrieve all the photos which stands for Jordan.

Pros: The prospect of automated organization and image searching by google search engine machine learning are smart that it's able to search out the photo that you want in just few seconds.

Cons: Search that is successfully found three scans of that image, still left out two others image when doing the same scanning simultaneously, as sometimes Google missed some of the proper tag and not able user to add in an own duplication tag in the same album without creating a new album.

2.1.3 IstockPhoto

The istockphoto mobile app is run by a highly reliable Getty Images Company. Many of the advertising companies and graphic designer like to search photo in stock due to their affordable subscription monthly plan or yearly plan and diverse variation of photos can be purchased in stock. The istockphoto website can be challenging to use by users because some user prefers to browse without typing any keyword in the search function. The search function is through typing keyword and is easy-going and inbuilt for the user to use.

Pros: Users are able to search photo through typing the general keyword of he/she want to find.

Cons: User cannot add, add tag they want while uploading the photos.

2.2 Tag Photo Management

2.2.1 Focus

Focus is an android platform application that able user to select tags from predefined library or create own custom tag. The tagging system able to add several tags in each photo it makes the system able to read radical details from the data of the photos.

Pros: Can let user create own customs keywords or tags or choose from the predefined library.

Cons: Need to pay to unlock all the custom adding tag features.

2.2.2 QuickPic

QuickPic able user to organize pictures and put all the pictures in a folder, and can view the pictures in a grid-view format or list-view format. It does not provide the tagging system that Focus had but it's totally free-of-charge.

Pros: The display of images can be chosen by the user either grid-view or list-view and it is free-of-charge.

Cons: User can't create custom keywords or tags or choose from the predefined library.

2.2.3 Google Photos

Google Photos can let the user search photo based on the specific subject that contain within a photo using image analysis (predefined Google Cloud Vision API) and provide a service that can automatically upload any photo to Google's Cloud.

Pros: It provides labelling services that able to detect the subject within photos and display out the labels.

Cons: User need to pay the money within a certain amount of image use per month.

2.3 Image recognition services without image compression

2.3.1 Google Vision API: Image Analysis as a Service

In the year of 2015, Google Vision API has launched a RESTful interface that able to provide speedily analyses about the image content. The interfaces hide all the complexity about continually growing of image processing algorithms and machine language learning models. As a result, overall system accuracy has been improved in these models especially for object detection.

At this moment, API first accepts base64-encoded image series as input. For future release, maybe will be integrated with Google Cloud Storage with the purpose of API requests do not require any image uploads and will substantially offer more fast invocation.

Pros: The API detection features (Safe_Search_Detection, Logo_Detection, Landmark_Detection, Face_Detection, Text_Detection, Label_Detection) enable user to search semantic keyword or images to annotate all together with a single uploads.

Cons: Respond time for API will be slow if using all the features concurrently and didn't provide any dynamic auto image resize in mobile.

2.3.2 Google Cloud Vision API Image Recognition Services

Consequently, Alpha Testing of Google Cloud Vision API can provide fast and accurate queries just only take milliseconds to display the image requested by the end-user. But the limitation is it will take longer processing to upload the images to cloud because the image size was large.

Pros: Google Cloud Vision API can perform fast and accurate queries images in just few milliseconds.

Cons: Google Cloud Vision API doesn't not provide any image compression which able to compress the image to the minimum best quality to upload to Cloud Vision for Image Recognition Services and increase the speed of image query processor.

2.4 Conclusion

From the literature review, I found that the existing mobile based applications are performing the similar features in searching image by keywords in local phone storage.

But what really makes this Eidetic Search mobile application unique from others are the objectives, it's satisfying the user demand by having tag photo management, which enables user to search, add, , delete keyword in photos and retrieve it in a more convenience, shortest duration and more user-friendly way. For instance, it will make thorough analyses and filter out the probability score and other detail information within the photo using Google Cloud Vision API Label Detection and categorize it finely by listing the photos from descending order.

By having this improved system, Eidetic Search with significantly improved features such as: By using the image resizer function, the mobile application able to reduce the image size to minimize the best quality of the photos that are needed to send to the cloud, to implement the request Google Vision API detection in doing label indexing and abilities to help user to save the maximum data cellular network while uploading photos to perform Google Cloud Vision API services.

In results, the development of this system is trying to overcome the issues that faced by the current system, so that the user can reduce time consuming, while searching photo in an efficient and effective way, tag photo management to help user to organize the label within photos, like to add, delete and using the image resizer will reduce the maximum data network cellular consumption of the user when sending the request to the Google Cloud Vision API.

Chapter 3 Methodology

3.1.1 Eidetic Search System Architecture

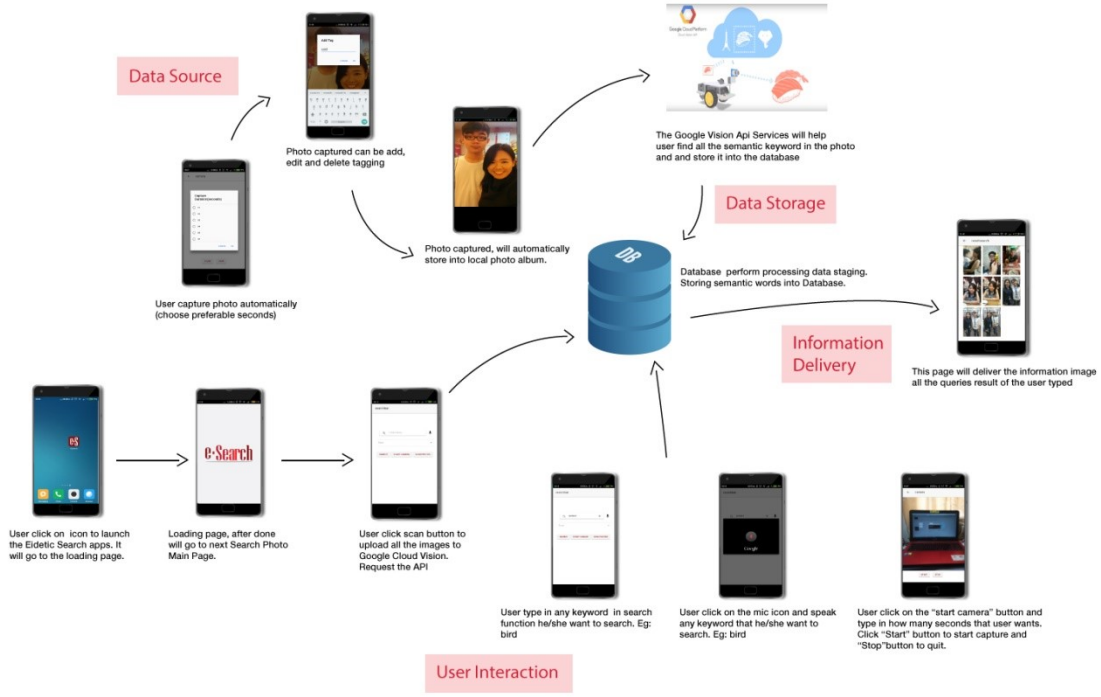


Figure 3.1 Eidetic Search System Architecture

3.1.2 Process User capture photo and store photo into database

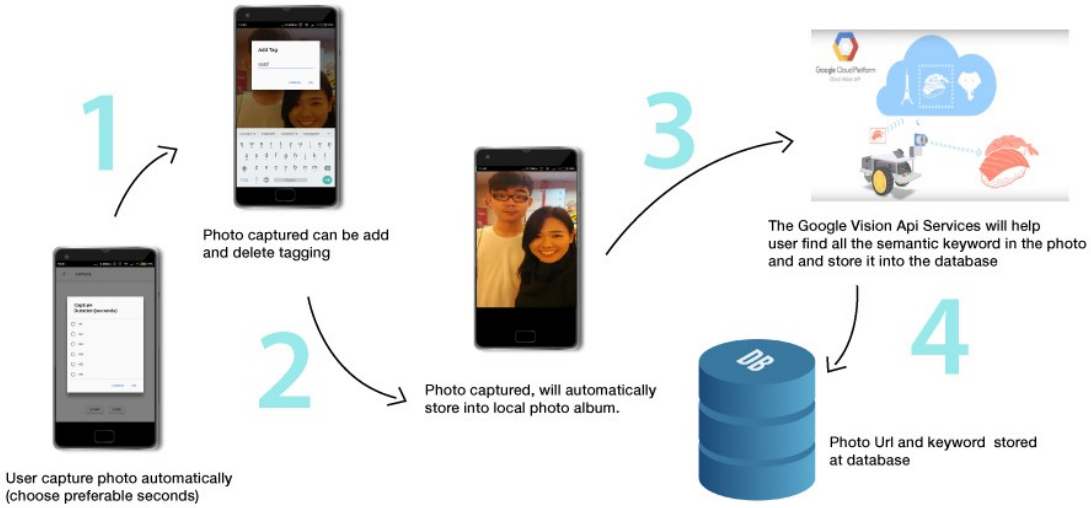


Figure 3.2 Process when user capture photo and store photo into database

3.1.3 Process User scan photos information into database for data storage

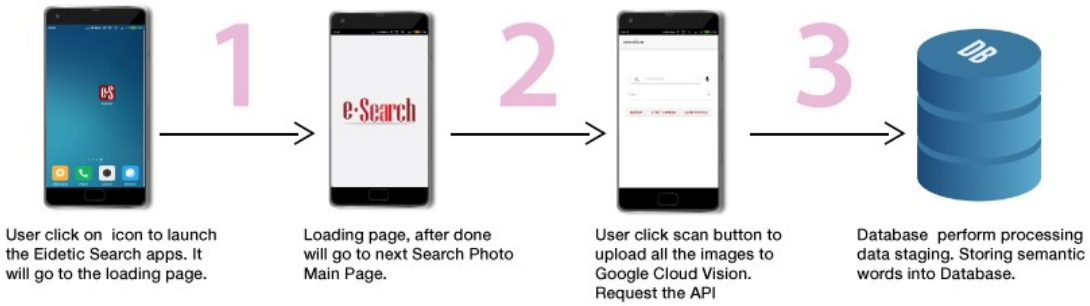


Figure 3.3 Process user scan photos information into database for data storage

3.1.4 Process User retrieving photo from database using Keyword

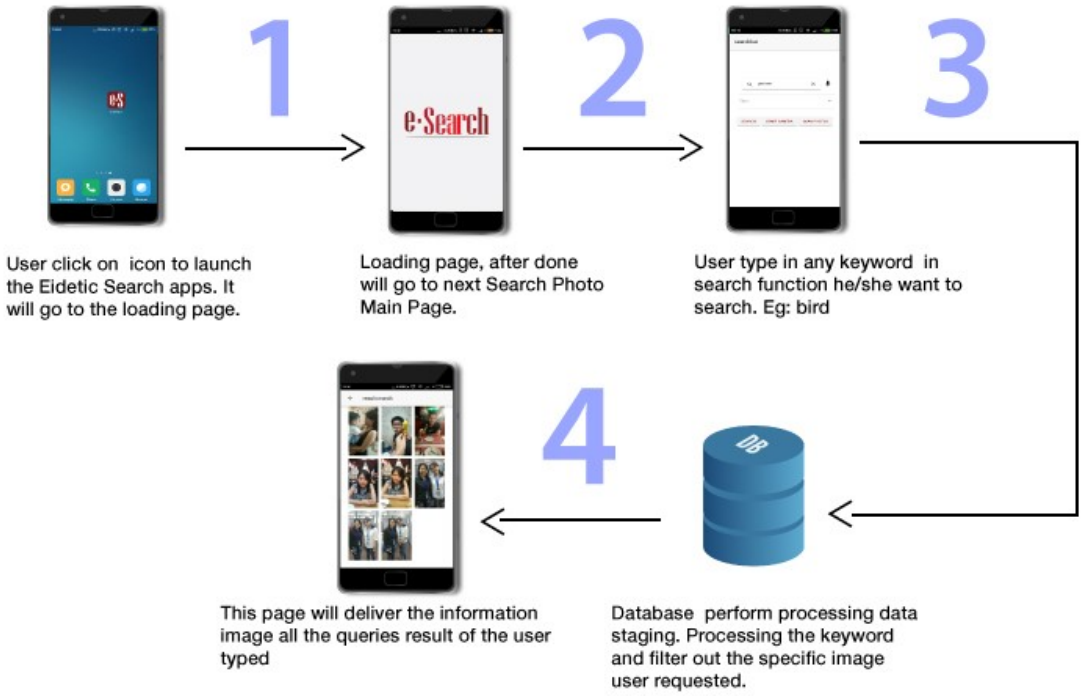


Figure 3.4 Process User retrieving photo from database using Keyword

3.1.5 Process User retrieving photo from database using Voice Recognition

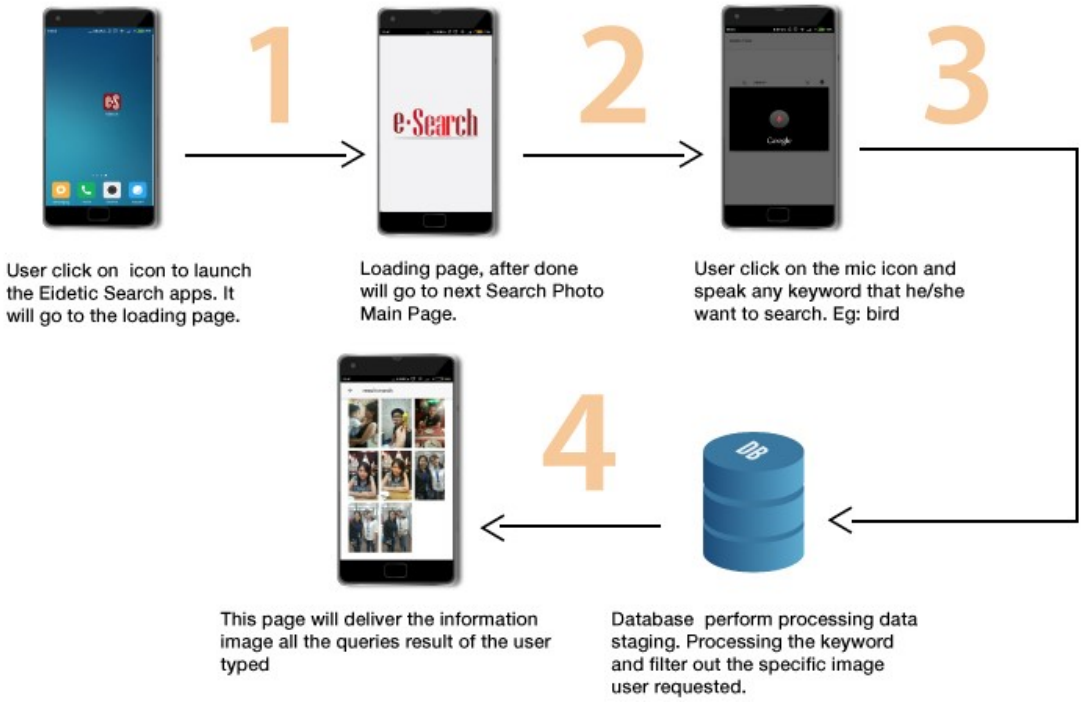


Figure 3.5 Process User retrieving photo from database using Voice Recognition

3.1.6 Process User performing Auto capture photo

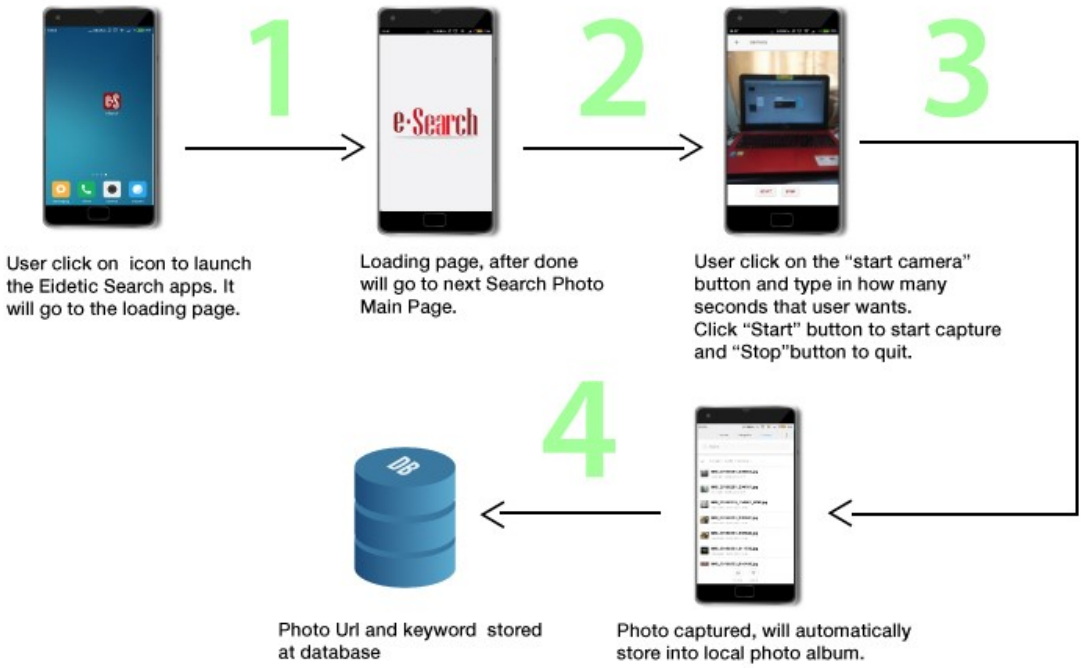


Figure 3.6 Process User performing Auto capture photo

3.2. Use Case Diagram

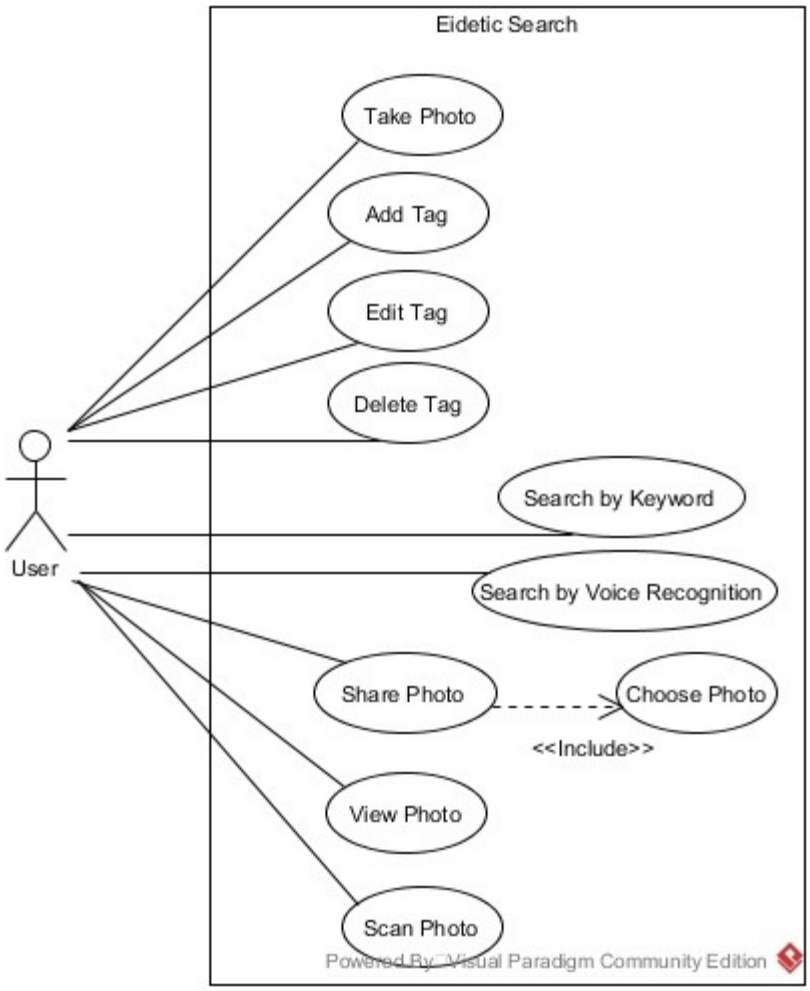


Figure 3.7 Use Case Diagram Eidetic Search

3.3 Activity Diagram

3.3.1 Activity Diagram Search Photo in Eidetic Search

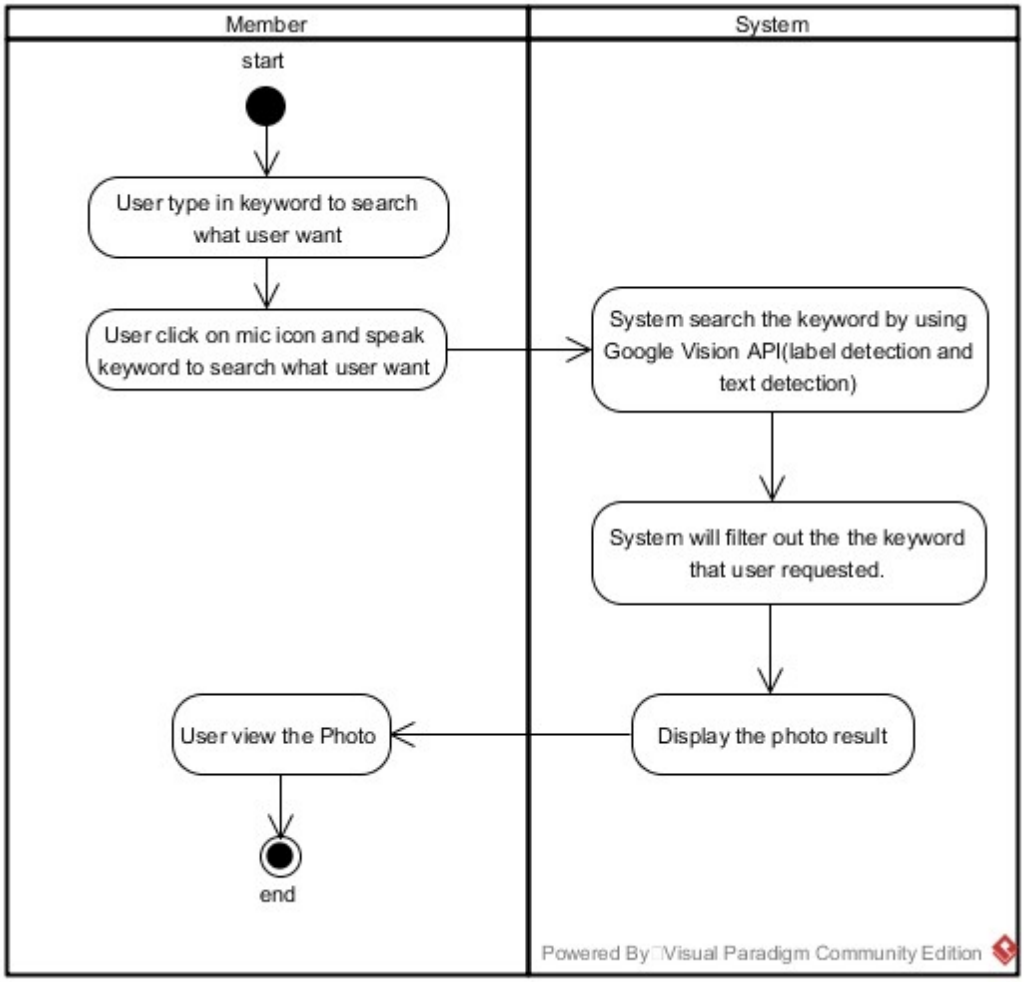


Figure 3.8 Activity Diagram Search Photo in Eidetic Search

3.3.2 Activity Diagram Take Photo in Eidetic Search

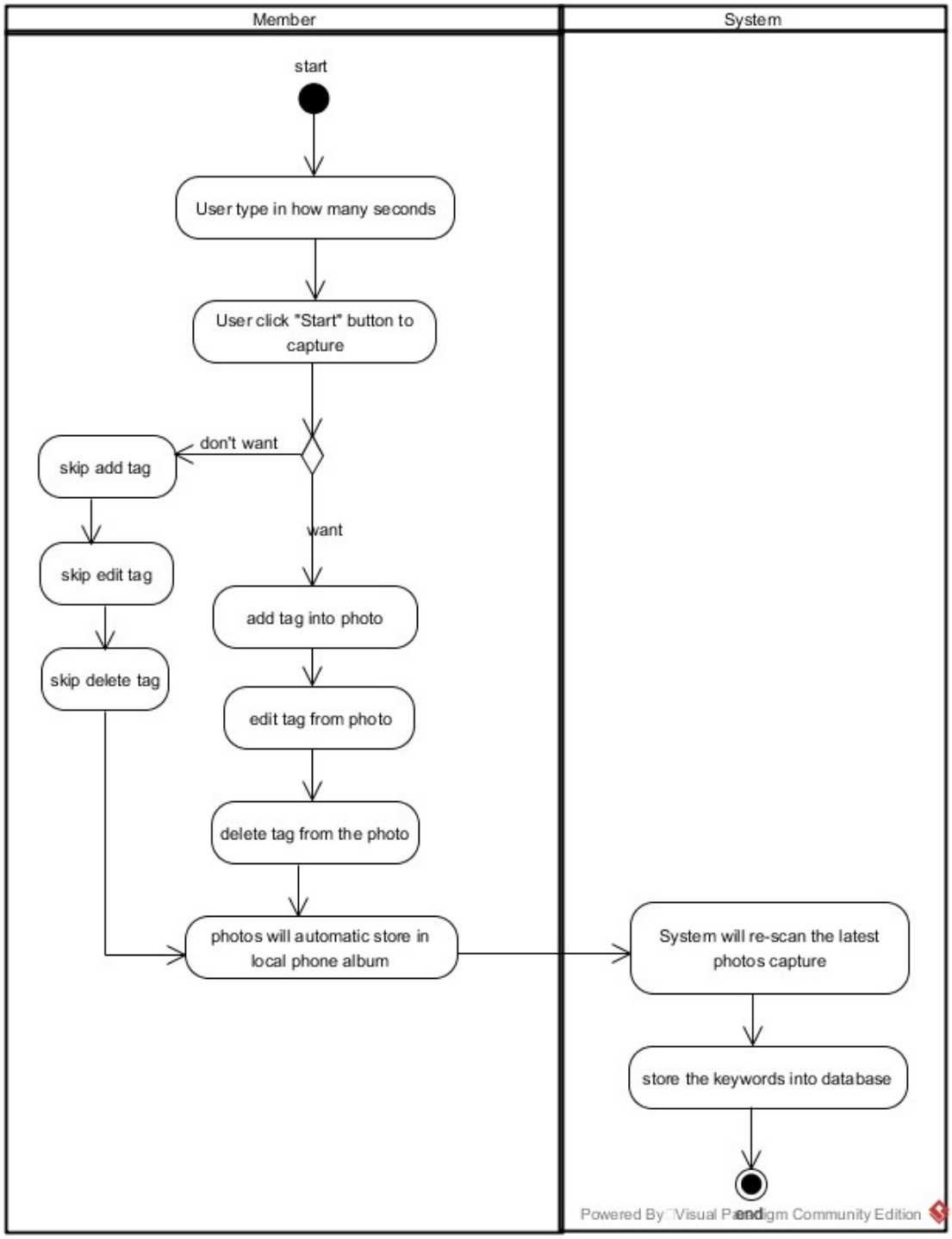


Figure 3.9 Activity Diagram Take Photo in Eidetic Search

3.3.3 Activity Diagram Share Photo in Eidetic Search

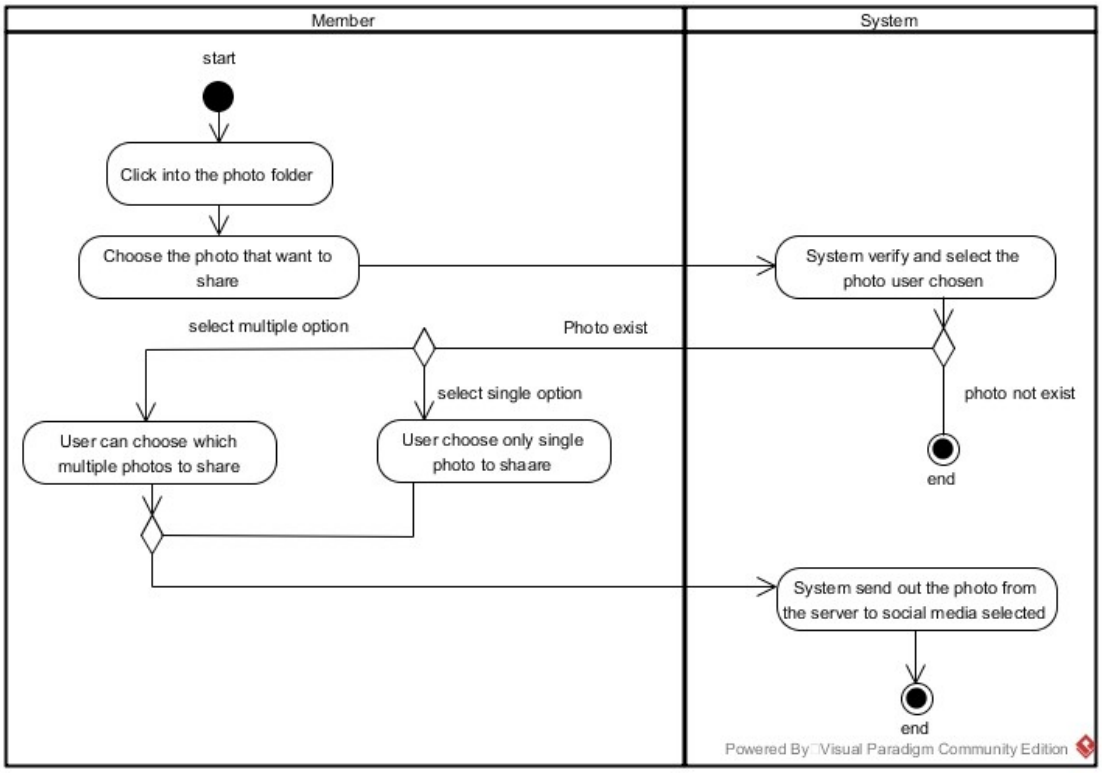


Figure 3.10 Activity Diagram Share Photo in Eidetic Search

3.4 Sequence Diagram

3.4.1 Sequence Diagram for Auto-Capturing Photo

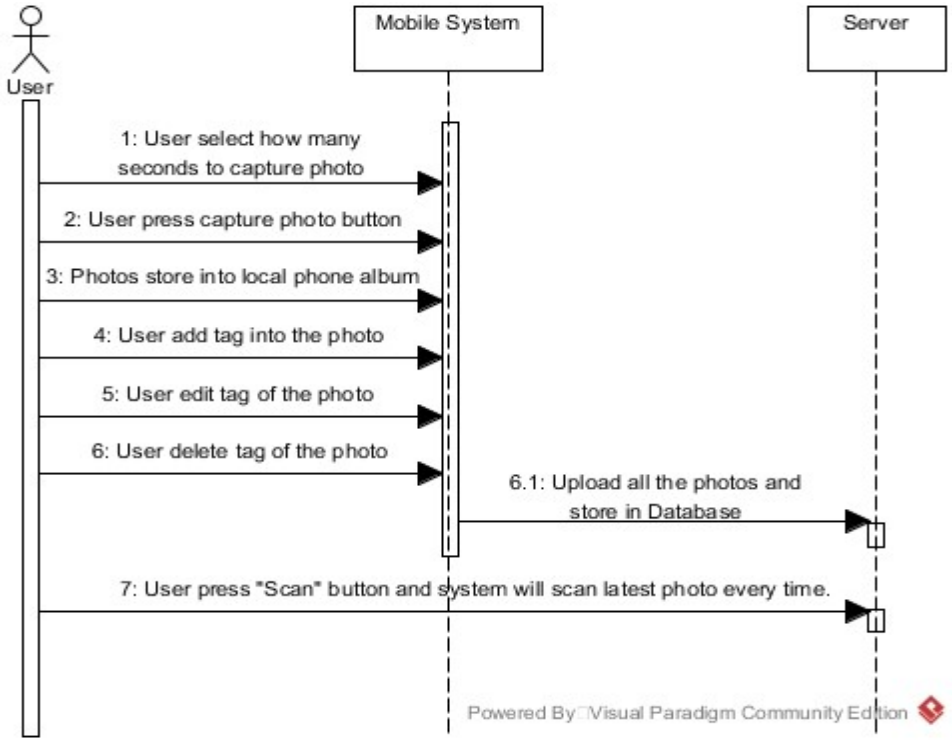


Figure 3.11 Sequence Diagram for Auto-Capturing Photo

3.4.2 Sequence Diagram for Search, Add, Delete Tag and View Photo

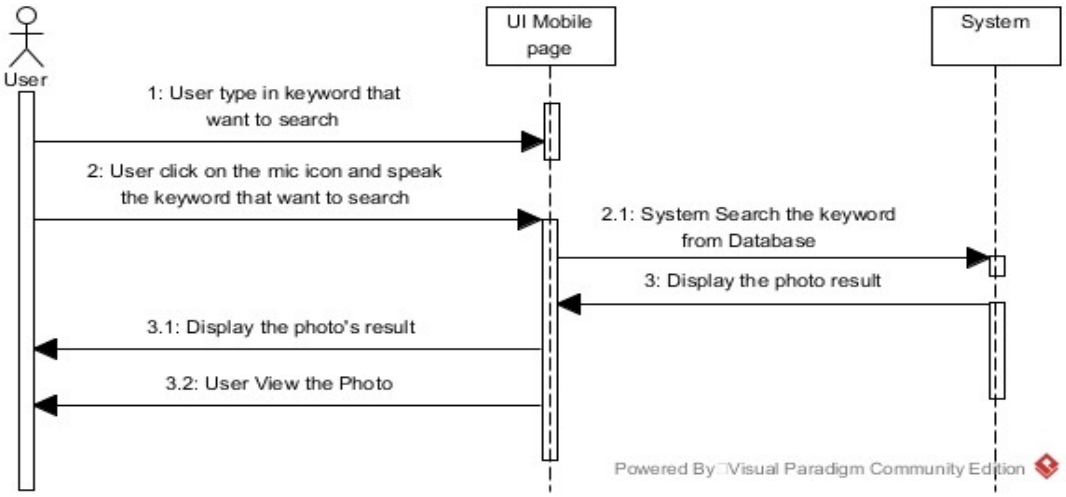


Figure 3.12 Sequence Diagram for Search, Add, Delete Tag and View Photo

3.4.3 Diagram for Share Photo to social media

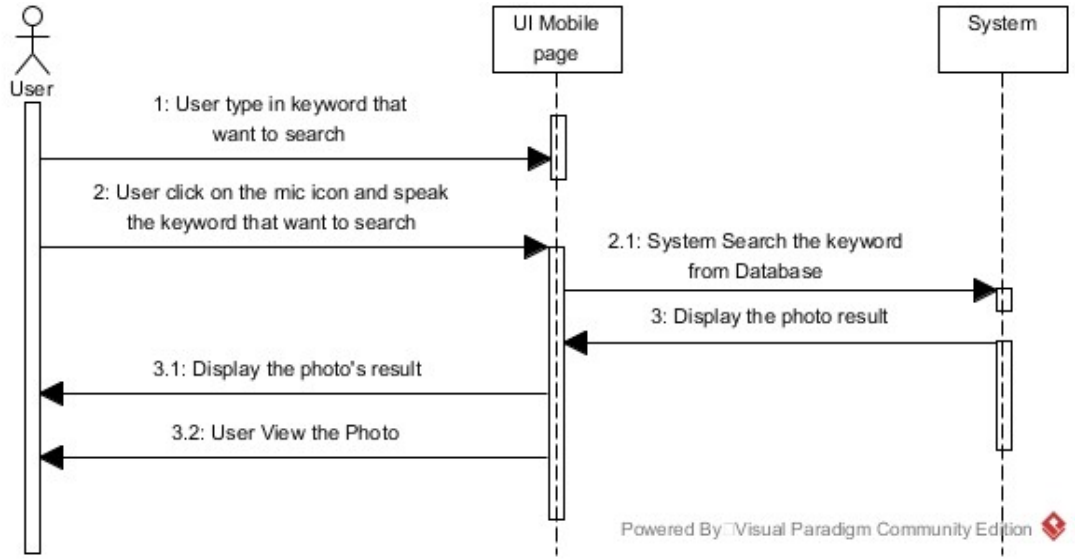


Figure 3.13 Diagram for Share Photo to social media

3.5 Class Diagram

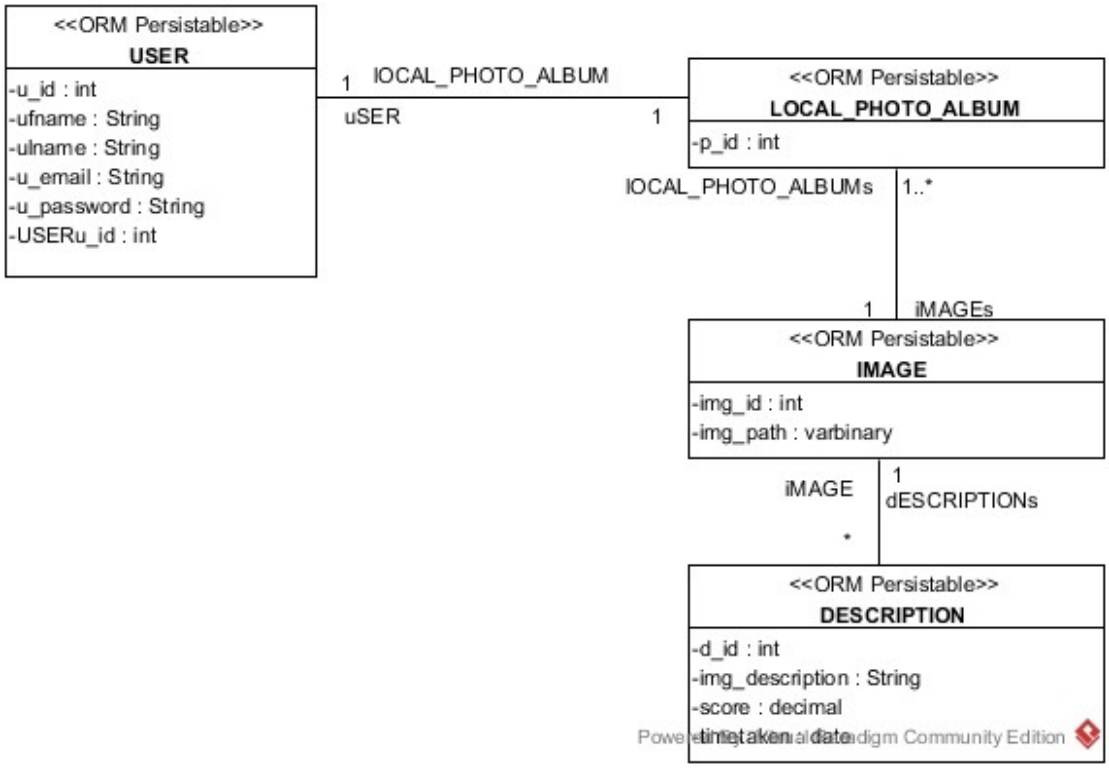


Figure 3.14 Class Diagram Eidetic Search

3.6 Entity-Relationship Diagram

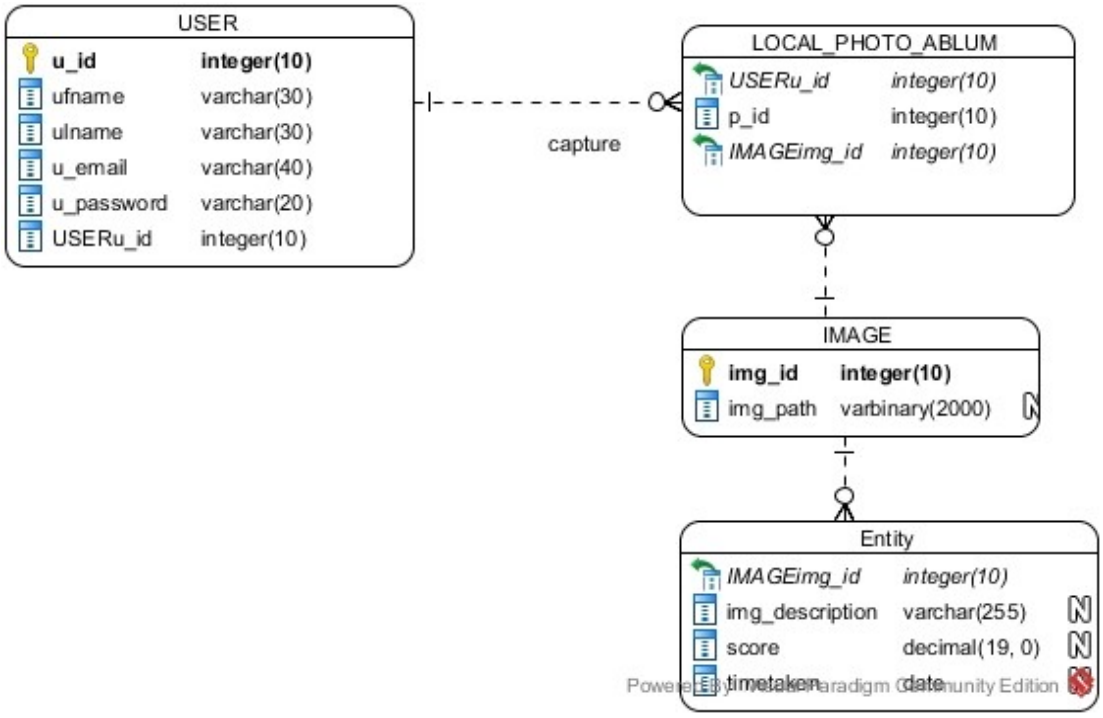


Figure 3.15 Entity Relational Diagram Eidetic Search

Chapter 4 – Implementation

In this mobile application based development, some of the implementations software's really need to be discussed. Currently, software's that is required for Eidetic Search Apps are Ionic2, Angular 2, NodeJs and PostgreSQL.

4.1 System Implementation and Requirement

Figure 4.1 System Implementation Eidetic Search

After doing some thorough analyses of the selection tools researches, Ionic 2 has been selected as Eidetic Search main development framework tool.Ionic 2 is a framework that able web developer to perform rapid app development for mobile web with distinct code base and all of the main app stores.

AtScript in Angular 2 is superset language that uses TypeScript's type of syntax to represent all possible types that can used to create runtime statement, instead of checking compile-time.

Flexible and minimal Express Node. j's are a development framework that uses for web application as it can provide a set of robust features for mobile and web applications. It uses uncountable HTTP middleware and function methods at discarding and creating a robust API, so that it is more easy and speedy for developer to use.

PostgreSQL is an open source object-relational database system and it is powerful because it has a solid history, development and proven architecture to provide data integrity, reliability, correctness and able to run on major Operating Systems.

4.1.2 Hardware Requirement

Phone

Model: RedMi Note2

Android 4.4.4 KitKat

MediaTek Helio X10 octa-core 64-bit processor

RAM: 2GB

Computer

System Type: 64-bit Operating System

Windows: Windows 10

Processor: Intel® Core™ i7-5500 @ 3 GHz

Memory: 4.00 GB

Input: Keyboard, Mouse

Display: Monitor

Documentation: Printer

4.1.3 Software Requirements

Back-end: Angular2, Node.js, PostgreSQL

Front-end: Ionic2, Bootstrap, Scss, Html

Interface: GUI (Graphical User Interface)

4.2 Implementation Code in Ionic2

4.2.1Google Cloud Vision API Application

In Eidetic Search Mobile Applications, the Google Cloud Vision API needs to be implemented into the source code. The developer uses it to perform Label Detection Service, to process all the photos in local phone photo album.

searchbar.ts

```
@Component({})
export class SearchbarPage {
  private GOOGLE_VISION_API_URL =
"https://vision.googleapis.com/v1/images:annotate?key=";
  private API_KEY: string = 'AIzaSyAi47rQVVLiZY1u_xQDvTCLdvvA2YKiMtM';
  }
```

4.2.2 Search by Keyword Function

This Search by Keyword is to let user to type in any keyword to filter out all the photos that are categorized in the same category based on the keyword typed and display it out.

search.html

```
<ion-content>
<ion- item>
<ion-searchbar (ionInput)="getItems($event)" [(ngModel)]="keyword" placeholder="Filter
Items"></ion-searchbar>
</ion-item>
<button ion-button color="light" (click)="presentLoadingDefault()">SearchPhoto</button>
</ion-content>
```

search.ts

```
presentLoadingDefault() {
    let loading = this.loadingCtrl.create({
      content: 'Please wait...',
```

```
    dismissOnPageChange: true
  });
  loading.present();
  setTimeout(() => {
    this.resultPhoto();
  }, 1500);
 }
```

4.2.3 Search by Speech Recognition Function

This Search by Voice Recognition is using Ionic2 Speech Recognition is to let user to speak any keyword to filter out all the photos that are categorize in the same category based on the keyword spoke out and display it out.

searchbar.html

```
<ion-content padding>
<ion-item>
                <ion-icon name="mic" (click)="speechRecord()" item-right></ion-icon>
        </ion-item>
</ion-content>
```

searchbar.ts

```
  SpeechRecognition.startListening(options)
      .subscribe(
      (matches: Array<string>) => {
        alert(matches[0]);
        this.keyword = matches[0];
        this.presentLoadingDefault();
      },
      (onerror) => alert('error:' + onerror)
      )
```

4.2.4 Auto Capture Function

The Auto Capture module is to let user to start a camera using Ionic 2 Framework and choose which second that are preferable for them to do the auto capturing and store all the photos into the local phone album.

camera.html

```
<ion-content padding>
        <div style="margin-top:145%">
        <button ion-button color="light" (click)="presentPrompt()">Start</button>
        <button ion-button color="light" (click)="stop()">Stop</button>
        </div>
</ion-content>
```

camera.ts

```
 stop() {
   if (this.start_had_click == true) {
     this.start_had_click = false;
     alert("Auto capture is stop");
     clearInterval(this.autoCapture);
   }
   else {
     alert("Auto capture did not start")
   }
 }

 presentPrompt() {
   CameraPreview.hide();
   let arr = Array(60);
   let temp;
   for (let i = 0; i < 60; i++) {
     if (i < 9) {
```

```
    temp = "0" + (i + 1).toString();
   }
   else temp = (i + 1).toString();
   let obj = {
    value: i + 1,
    type: 'radio',
    checked: false,
    label: temp
   };
   arr[i] = obj;
  }
```

4.2.5 Display Photo Function

This Display Photos Module is to display all the photos searched by the user using keyword and speech recognition, keyword in a dynamic gird view with an 2D-ArrayList [9] [3].

research.html

```
<ion-content padding>
        <ion-grid>
                <ion-row wrap responsive *ngFor="let row of grid">
                        <ion-col width-35 *ngFor="let file_uri of row">
                                <ion-thumbnail>
<img [src]="file_uri.resized_path" (click)='view_photo(file_uri)'
alt="http://www.drodd.com/images14/white6.png">
                                </ion-thumbnail>
                        </ion-col>
                </ion-row>
        </ion-grid>
</ion-content>
```

research.ts

```
import { ViewphotoPage } from '../viewphoto/viewphoto';
```

```
export class ResultsearchPage {
  public images_original: any = [];
  public images_resized: any = [];

  public grid: any;
  private base_url: any = "http://192.168.1.36:3000";
  private keyword = "";
  private range: any;

  constructor(public navCtrl: NavController, public navParams: NavParams, public http: Http) {
    this.grid = Array(Math.ceil(this.images_original.length / 3));
  }

  get_result_find_by_id() {
    //alert(this.keyword);
    var params = {
      keyword: this.keyword,
      range: this.range
    };

    this.http.post(this.base_url + '/find_by_keyword_with_range/', body, { headers: head })/*save photo info into database*/
      .map(res => res.json())
      .subscribe(res => {
        //alert(JSON.stringify(res));
        //console.log(res);
        if (res.rowCount == 0) {
          alert("No images found...");
        }
        else {
          this.images_original = res.rows;
```

```
        for (let i = 0; i < this.images_original.length; i++) {
          this.imageResize(this.images_original[i].img_path, i);
        }
      }
    });
  }

  display_images() {
    let rowNum = 0; //counter to iterate over the rows in the grid
    for (let i = 0; i < this.images_resized.length; i += 3) { //iterate images
      let img_path = {
        original_path: null,
        resized_path: "http://www.drodd.com/images14/white6.png"
      };

      this.grid[rowNum] = Array(3); //declare two elements per row

      if (this.images_resized[i]) { //check file URI exists
        let img_path = {
          original_path: this.images_original[i].img_path,
          resized_path: this.images_resized[i]
        };

        //alert(i + " : " + img_path.resized_path)
        this.grid[rowNum][0] = img_path; //insert image
      }
  }
```

4.2.6 Image Resizer function

The Image Resizer function is help user to reduce the high resolution image to the best image minimum size when user request for label detection.

Search.ts

```
import { ImageResizer, ImageResizerOptions, SpeechRecognition } from 'ionic-native';

declare var options: any;
imageResize(uri, index) {
  let options = {
    uri: uri,
    folderName: 'Camera',
    quality: 80,
    width: 1000,
    height: 1000
  } as ImageResizerOptions;

  ImageResizer
    .resize(options)
    .then((filePath: string) => {
      //alert(filePath);
      //alert("resize done");
      this.images_resized[index] = filePath;
      if (index == this.images_original.length - 1) {
        this.display_images();
      }
    })
    .catch((err) => {
      alert(err);
      console.log('Error occured');
    });
}
```

4.2.7 Share media Function

The Share Media feature is to let user to share the image taken to any social media in the mobile application such as Facebook, Instagram and Whatapps.

resultsearch.ts

```
shareMedia(select) {
  if (select == 'facebook') {
    // Share via fb
    SocialSharing.shareViaFacebook('Body', this.image.original_path,
this.image.original_path).then(() => {
      // Success!
      //alert('successfully facebook')
    }).catch((err) => {
      // Error!
      alert(err)
    });
  }
  else if (select == 'instagram') {
    SocialSharing.shareViaInstagram('Body', this.image.original_path).then(() => {
      // Success!
      //alert('Successfully Instragram')
    }).catch((err) => {
      // Error!
      alert(err)
    });
  }
  else if (select == 'whatapps') {
    SocialSharing.shareViaWhatsApp('', this.image.original_path, '').then(() => {
      // Success!
      //alert('successfully whatapps')
    }).catch((err) => {
```

```
      // Error!
      alert(err)
    });
  }
}

socialShareList() {
  let alerts = this.alertCtrl.create({
    title: 'Share Media',
    inputs: [
      {
        value: 'facebook',
        type: 'radio',
        label: 'Facebook',
        checked: false
      },
      {
        value: 'whatapps',
        type: 'radio',
        label: 'Whatapps',
        checked: false
      },
       {
        value: 'instagram',
        type: 'radio',
        label: 'Instagram',
        checked: false
      }
    ],
    buttons: [
      {
```

```
        text: 'Cancel',
        role: 'cancel',
        handler: data => {
          console.log('Cancel clicked');
        }
      },
      {
        text: 'OK',
        handler: data => {
          //alert(data);
          this.shareMedia(data);
          console.log('OK clicked');
        }
      }
    ]
  });
  alerts.present();
 }
```

4.2.8 Filter to Search Photo Function

The filter feature is able to let user to do an advance searching for the photos based on the duration that has been given such as by 1 hour, 1 day, 1 week, 1 month or all time. It is use to reduce the time searching to be faster than scrolling the photos one by one.

searchbar.ts

```
resultPhoto() {
  //alert(this.keyword);
  //alert(this.time);
  let range;
  if (this.time == "all") {
    range = 0;
```

```
  } else if (this.time == "one_hour") {
    range = -1;
  } else if (this.time == "one_day") {
    range = 1;
  } else if (this.time == "one_week") {
    range = 7;
  } else if (this.time == "one month") {
    range = 30;
  }
  this.navCtrl.push(ResultsearchPage, { keyword: this.keyword, range: range });
 }
```

4.2.9 Add Tag Function

The Add tag feature is able to let user to add own keyword within the photo. As not all of the description that google API detection is suitable for the user to use it. For instance, the photo user took has some fries and nugget and the label detection show to user is food. But what user really want is nugget or fries, so user can add in own keywords within the photos to make the search function more efficient.

viewphoto.ts

```
addTag() {
  let alert = this.alertCtrl.create({
    title: 'Add Tag',
    inputs: [
      {
        name: 'tag',
        placeholder: 'Tag'
      }
    ],
    buttons: [
      {
        text: 'Cancel',
```

```
        role: 'cancel',
        handler: data => {
          console.log('Cancel clicked');
        }
      },
      {
        text: 'OK',
        handler: data => {
          let navTransition = alert.dismiss();
          navTransition.then(() => {
            this.confirmAddTag(data.tag);
          });
        }
      }
    ]
  });
  alert.present();
}
 confirmAddTag(tag_name) {
  let params =
    {
      path: this.image.original_path,
      tag: tag_name
    };

  let body = JSON.stringify(params);
  let head = new Headers({
    'Content-Type': 'application/json'
  });


  this.http.post(this.base_url + '/insert/addtag', body, { headers: head })/*save photo info
into database*/
```

```
    .map(res => res)
    .subscribe(data => {
      console.log(data);
      alert("Add Successful");
    });
  }
  getAllTag() {
    let params =
    {
      path: this.image.original_path,
    };

    let body = JSON.stringify(params);
    let head = new Headers({
      'Content-Type': 'application/json'
    });

    this.http.post(this.base_url + '/tag/getalltag', body, { headers: head })/*save photo info into database*/
    .map(res => res.json())
    .subscribe(res => {
      console.log(res);
      let arr = Array(res.rowCount);
      for (let i = 0; i < res.rows.length; i++) {

        let obj = {
          value: res.rows[i].img_desc,
          type: 'checkbox',
          checked: false,
          label: res.rows[i].img_desc
        };
```

```
        arr[i] = obj;
      }
      this.presentPrompt(arr);
    });
  }

  presentPrompt(arr) {
    let alert = this.alertCtrl.create({
      title: 'Delete Tag',
      inputs: arr,
      buttons: [
        {
          text: 'Cancel',
          role: 'cancel',
          handler: data => {
            console.log('Cancel clicked');
          }
        },
        {
          text: 'OK',
          handler: data => {
            console.log(data);
            this.deleteTag(data);
            console.log('OK clicked');
          }
        }
      ]
    });
    alert.present();
  }
```

4.2.10 Delete Tag Function

The Delete tag feature is able to let user to delete the keywords that has not been used or not necessary for the photo itself.

Viewphoto.ts

```
deleteTag(data) {
  let alert = this.alertCtrl.create({
    title: 'Confirm Delete',
    message: 'Do you want to delete this tag?',
    buttons: [
      {
        text: 'Cancel',
        role: 'cancel',
        handler: () => {
          console.log('Cancel clicked');
        }
      },
      {
        text: 'OK',
        handler: () => {
          console.log('OK clicked');
          let navTransition = alert.dismiss();
          navTransition.then(() => {
            data.forEach((item) => {
              this.confirmDeleteTag(item);
            });
          });
        }
      }
    ]
  });
```

```
  alert.present();
 }

 confirmDeleteTag(tag) {
  let params =
   {
    path: this.image.original_path,
    tag: tag
   };

  let body = JSON.stringify(params);
  let head = new Headers({
   'Content-Type': 'application/json'
  });

  this.http.post(this.base_url + '/delete/deletetag', body, { headers: head })/*save photo info into database*/
   .map(res => res)
   .subscribe(data => {
    console.log(data);
    //alert("Delete Successful");
    //this.navCtrl.setRoot(SearchbarPage);
   });
 }
```

4.2.11 Scan Photos Function

The Scan Photo Button is let user to scan all the photos from the local photo album and store it to the database. Each time there is some new photos taken, user can press the button to scan the new photos and store it into the database. The function will check if the photos already exist it won't scan again or vice-versa.

searchbar.html

```
<button ion-button class="btn" color="light" (click)="initializeItems()">Scan Photos</button>
```

searchbar.ts

```
initializeItems() {
   this.after = [];
   this.http.get(this.base_url + '/images')
    .map(res => res.json())
    .subscribe(res => {
      //alert(JSON.stringify(res.rows[0].img_path));
      alert(res.rowCount);
      this.before = res.rows;
      this.listDirectory2();
      if (res.rowCount != 0) {
        //this.database_list = res.rows;
      }
    });
 }
```

4.3 Implementation Code in Node.js

4.3.1 Add Tag Function

The add tag function in Node.js is a server code that act as a middle bridge that connect what user do in in Ionic 2 and perform adding tag process to the database.

```
app.post('/insert/addtag', function (req, res) {
   //pgconnect
   pg.connect(connect, function (err, client, done) {
      if (err) {
         return console.error('error fetching client from pool', err);
```

```
    }
    client.query('INSERT INTO descriptions(img_desc, score, img_id) VALUES($1, 1.0, (SELECT img_id from image WHERE img_path=$2))', [req.body.tag, req.body.path], function (err, result) {
    });
    res.send(res.json);
    done();
  });
});
```

4.3.2 Delete Tag Function

The delete tag function in Node.js is a server code that act as a middle bridge that connect what user do in in Ionic 2 and perform deleting tag process to the database.

```
app.post('/delete/deletetag', function (req, res) {
  //pgconnect
  pg.connect(connect, function (err, client, done) {
    if (err) {
      return console.error('error fetching client from pool', err);
    }
    client.query('DELETE FROM descriptions WHERE img_id=(SELECT img_id from image WHERE img_path=$1) AND score=1.0 AND img_desc=$2;', [req.body.path, req.body.tag], function (err, result) {
    });
    res.send(res.json);
    done();
  });
});
```

4.3.3 Filter Search by keyword and time Function

The filter search by keyword and time function in Node.js is a server code that act as a middle bridge that connect what user do in in Ionic 2 and perform advance search based on time taken process to the database.

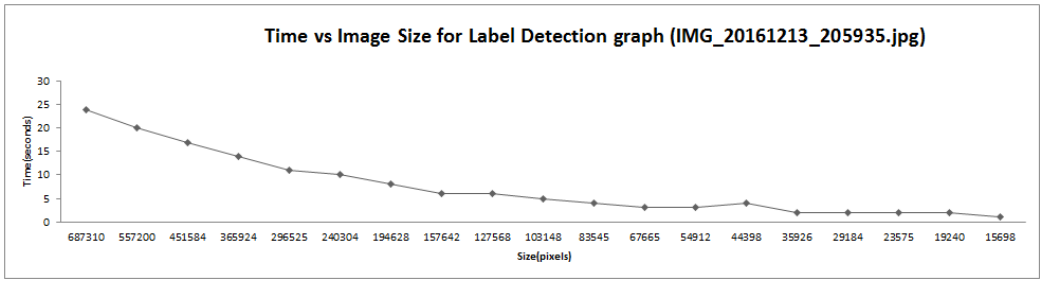
```
app.post('/find_by_keyword_with_range', function (req, res) {
    let range;
    let query;
    range = req.body.range;
    if (req.body.range == -1) {
        query = '(SELECT EXTRACT(HOURS FROM (now() - timetaken))>=$2)';
        range = 1;
    } else {
        query = '(SELECT EXTRACT(DAYS FROM (now() - timetaken))>=$2)';
    }
    //pgconnect
    pg.connect(connect, function (err, client, done) {
        if (err) {
            return console.error('error fetching client from pool', err);
        }
        client.query('SELECT img_path FROM image WHERE img_id IN (SELECT img_id
FROM descriptions WHERE img_desc=$1 AND score>0.9 AND ' + query + ')',
[req.body.keyword, range], function (err, result) {
            if (err) {
                return console.error('error running query', err);
            }
            res.send(result);
            done();
        });}
```

Chapter 5 Experimental Results

5.1. Comparison Time vs Image Size for Label Detection graph

Essentially Google Cloud Vision API able to detect the common label objects within image in a very high accuracy even on any image sizes. As if the image has more common or more general attributes will be more accurate to be display out. But the result that we want is to show the best minimum image size to detect the most label detection of an image that we want.

5.1.1 Time Vs Image Size for Label Detection graph (IMG_20161213_205935.jpg)



IMG_20161213_205935.jpg

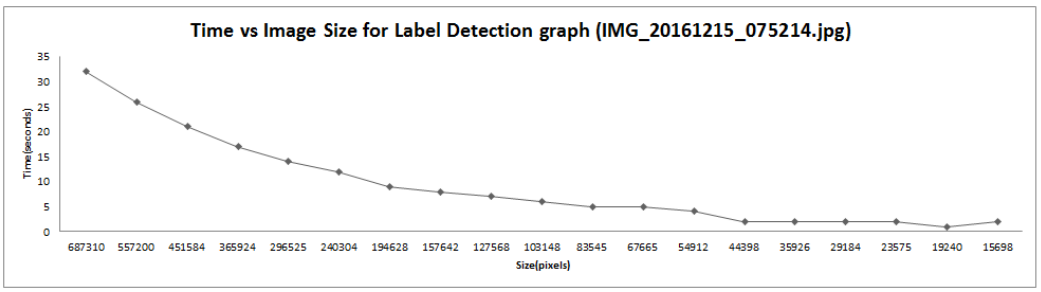
#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	622 x 1105	622	1105	687310	24	ice cream	0.91886324
2	560 x 995	560	995	557200	20	ice cream	0.9190913
3	504 x 896	504	896	451584	17	ice cream	0.9190102
4	454 x 806	454	806	365924	14	ice cream	0.91642684
5	409 x 725	409	725	296525	11	ice cream	0.92357814
6	368 x 653	368	653	240304	10	ice cream	0.92112947
7	331 x 588	331	588	194628	8	ice cream	0.925873
8	298 x 529	298	529	157642	6	ice cream	0.9222899
9	268 x 476	268	476	127568	6	ice cream	0.9263039
10	241 x 428	241	428	103148	5	ice cream	0.9202181
11	217 x 385	217	385	83545	4	ice cream	0.92604685
12	195 x 347	195	347	67665	3	ice cream	0.9235902
13	176 x 312	176	312	54912	3	ice cream	0.91228884
14	158 x 281	158	281	44398	4	ice cream	0.9125753
16	128 x 228	128	228	29184	2	food	0.9140951
17	115 x 205	115	205	23575	2	food	0.8810265
18	104 x 185	104	185	19240	2	food	0.88628435
19	94 x 167	94	167	15698	1	food	0.83071256

Figure 5.1 Label Detection for IMG_20161213_205935.jpg

The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *ice-cream* word will display in their mind

first. Therefore, the minimum best image size display is in *142px x 253px* and *ice-cream* description is fall on *2* seconds with the score *0.89094764.*

Figure 5.1.2 Time Vs Image Size for Label Detection graph for (IMG_20161215_075214.jpg)



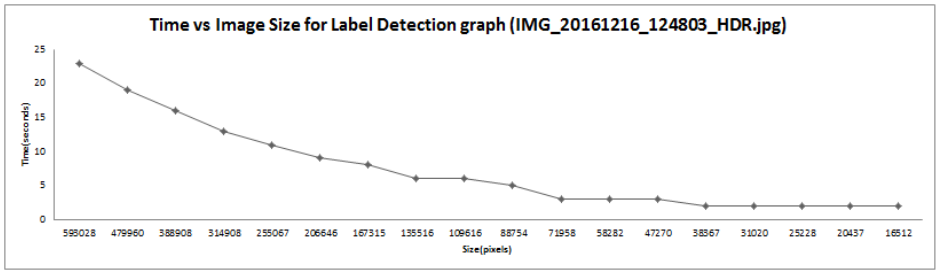
IMG_20161215_075214.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	622 x 1105	622	1105	687310	32	face	0.9741372
2	560 x 995	560	995	557200	26	face	0.97353745
3	504 x 896	504	896	451584	21	face	0.9722727
4	454 x 806	454	806	365924	17	face	0.9743778
5	409 x 725	409	725	296525	14	face	0.97433877
6	368 x 653	368	653	240304	12	face	0.97312194
7	331 x 588	331	588	194628	9	face	0.9728611
8	298 x 529	298	529	157642	8	face	0.97239995
9	268 x 476	268	476	127568	7	face	0.97287
10	241 x 428	241	428	103148	6	face	0.97409075
11	217 x 385	217	385	83545	5	face	0.97133756
12	195 x 347	195	347	67665	5	face	0.97162026
13	176 x 312	176	312	54912	4	face	0.9676548
14	158 x 281	158	281	44398	2	face	0.97070825
15	142 x 253	142	253	35926	2	face	0.964757
16	128 x 228	128	228	29184	2	face	0.9671888
17	115 x 205	115	205	23575	2	face	0.96656495
19	94 x 167	94	167	15698	2	face	0.9552888

Figure 5.2 Label Detection for IMG_20161215_075214.jpg

The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *face* word will display in their mind first. Therefore, the minimum best image size display is in *104px x 185px* and *face* description is fall on *1* seconds with the score *0.9615646.*

5.1.3 Time Vs Image Size for Label Detection graph (IMG_20161216_124803_HDR.jpg)



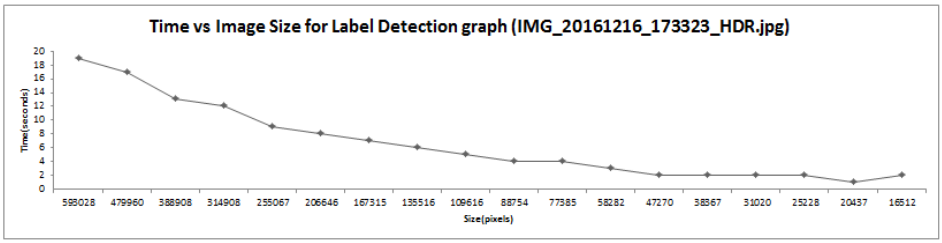
IMG_20161216_124803_HDR.jpg

#	Image Size	Width	Height	Size(pixels)	Time(second	Description	Score
1	1026 x 578	1026	578	593028	23	food	0.95433414
2	923 x 520	923	520	479960	19	food	0.95492077
3	831 x 468	831	468	388908	16	food	0.95346576
4	748 x 421	748	421	314908	13	food	0.9534016
5	673 x 379	673	379	255067	11	food	0.95442516
6	606 x 341	606	341	206646	9	food	0.9524497
7	545 x 307	545	307	167315	8	food	0.95384985
8	491 x 276	491	276	135516	6	food	0.9531134
9	442 x 248	442	248	109616	6	food	0.95335585
10	398 x 223	398	223	88754	5	food	0.95010644
11	358 x 201	358	201	71958	3	food	0.951935
12	322 x 181	322	181	58282	3	food	0.9509136
13	290 x 163	290	163	47270	3	food	0.94855845
14	261 x 147	261	147	38367	2	food	0.948405
15	235 x 132	235	132	31020	2	food	0.94969094
16	212 x 119	212	119	25228	2	food	0.9529761
17	191 x 107	191	107	20437	2	food	0.95018196

Figure 5.3 Label Detection for IMG_20161216_124803_HDR.jpg

The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the burger or fries word will display in their mind first. But the label detection detected food description. Therefore, the minimum best image size display is in 172px x 96px and food description is fall on 1 seconds with the score 0.95489347.

5.1.3 Time Vs Image Size for Label Detection graph (IMG_20161216_173323_HDR.jpg)



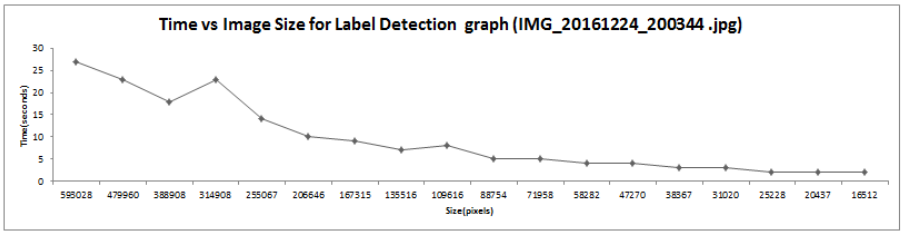
IMG_20161216_173323_HDR.jpg

	Image Size	Width	Height	Size(pixels)	ime(Seconds	Description	Score
1	1026 x 578	1026	578	593028	19	airliner	0.9537918
2	923 x 520	923	520	479960	17	airliner	0.9559838
3	831 x 468	831	468	388908	13	airliner	0.95556074
4	748 x 421	748	421	314908	12	airliner	0.954089
5	673 x 379	673	379	255067	9	airline	0.9554799
6	606 x 341	606	341	206646	8	airline	0.9573363
7	545 x 307	545	307	167315	7	airline	0.9592908
8	491 x 276	491	276	135516	6	airline	0.96041864
9	442 x 248	442	248	109616	5	airline	0.95914996
10	398 x 223	398	223	88754	4	airliner	0.9519308
11	358 x 201	385	201	77385	4	airline	0.9553834
12	322 x 181	322	181	58282	3	airline	0.96010494
13	290 x 163	290	163	47270	2	airline	0.95772165
14	261 x 147	261	147	38367	2	airline	0.9610022
15	235 x 132	235	132	31020	2	airline	0.9620477
16	212 x 119	212	119	25228	2	airline	0.94250697
18	172 x 96	172	96	16512	2	airliner	0.89643025

Figure 5.4 Label Detection for IMG_20161216_173323_HDR.jpg

The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the burger or fries word will display in their mind first. But the label detection detected food description. Therefore, the minimum best image size display is in 172px x 96px and airline description is fall on 1 seconds with the score 0.95489347.

5.1.4 Time Vs Image Size for Label Detection graph (IMG_20161224_200344.jpg)



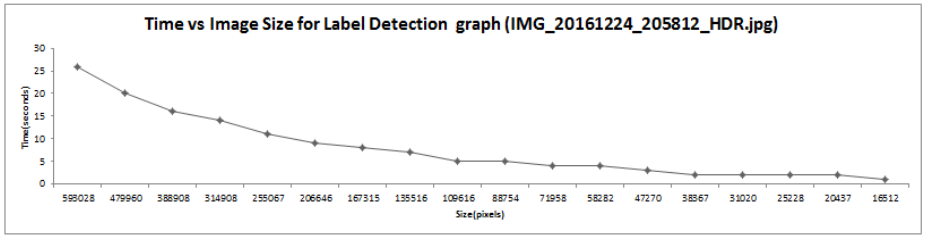
IMG_20161224_200344.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	1026 x 578	1026	578	593028	27	metropolitan area	0.89671844
2	923 x 520	923	520	479960	23	metropolitan area	0.90177286
3	831 x 468	831	468	388908	18	metropolitan area	0.896159
4	748 x 421	748	421	314908	23	metropolitan area	0.93477964
5	673 x 379	673	379	255067	14	metropolitan area	0.87218434
6	606 x 341	606	341	206646	10	metropolitan area	0.86588675
7	545 x 307	545	307	167315	9	night	0.8482648
8	491 x 276	491	276	135516	7	night	0.8493188
9	442 x 248	442	248	109616	8	landmark	0.8447707
10	398 x 223	398	223	88754	5	christmas lights	0.83731574
11	358 x 201	358	201	71958	5	christmas lights	0.842682
12	322 x 181	322	181	58282	4	christmas lights	0.8620049
13	290 x 163	290	163	47270	4	christmas lights	0.8618988
14	261 x 147	261	147	38367	3	christmas lights	0.86780363
15	235 x 132	235	132	31020	3	christmas lights	0.8747766
16	212 x 119	212	119	25228	2	christmas lights	0.8754165
17	191 x 107	191	107	20437	2	christmas lights	0.8894585

Figure 5.5 Label Detection for IMG_20161224_200344.jpg

The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *Christmas tree* word will display in their mind first. But the label detection detected *Christmas lights* description. Therefore, the minimum best image size display is in *172px x 96px* and Christmas light description is fall on *2* seconds with the score *0.88195306.*

5.1.5 Time Vs Image Size for Label Detection graph (IMG_20161224_205812_HDR.jpg)



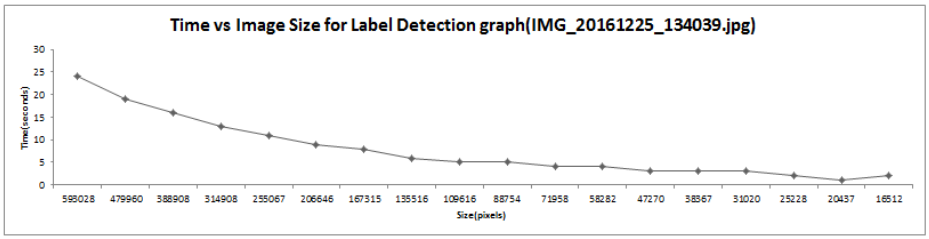
IMG_20161224_205812_HDR.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	1026 x 578	1026	578	593028	26	night	0.8696515
2	923 x 520	923	520	479960	20	night	0.87143135
3	831 x 468	831	468	388908	16	night	0.8626419
4	748 x 421	748	421	314908	14	night	0.8591858
5	673 x 379	673	379	255067	11	night	0.865366
6	606 x 341	606	341	206646	9	night	0.8685078
7	545 x 307	545	307	167315	8	night	0.8579482
8	491 x 276	491	276	135516	7	landmark	0.83433723
9	442 x 248	442	248	109616	5	night	0.8160931
10	398 x 223	398	223	88754	5	night	0.7538067
11	358 x 201	358	201	71958	4	plaza	0.5861907
12	322 x 181	322	181	58282	4	christmas lights	0.53189427
13	290 x 163	290	163	47270	3	christmas decoration	0.5657912
14	261 x 147	261	147	38367	2	christmas lights	0.5136988
15	235 x 132	235	132	31020	2	christmas decoration	0.55361855
16	212 x 119	212	119	25228	2	christmas lights	0.5851601
17	191 x 107	191	107	20437	2	christmas decoration	0.7358428

Figure 5.6 Label Detection for IMG_20161224_205812_HDR.jpg

The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *heritage building* word will display in their mind first. But the label detection detected *Christmas decoration* description. Therefore, the minimum best image size display is in *172px x 96px* and Christmas decoration description is fall on *1* seconds with the score *0.7662213*.

5.1.5 Time Vs Image Size for Label Detection graph (IMG_20161225_134039.jpg)



IMG_20161225_134039.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Secon	Description	Score
1	1026 x 578	1026	578	593028	24	green	0.93538237
2	923 x 520	923	520	479960	19	green	0.9304343
3	831 x 468	831	468	388908	16	green	0.9343031
4	748 x 421	748	421	314908	13	green	0.92323303
5	673 x 379	673	379	255067	11	green	0.9240687
6	606 x 341	606	341	206646	9	green	0.92584044
7	545 x 307	545	307	167315	8	green	0.91922164
8	491 x 276	491	276	135516	6	plant	0.9059518
9	442 x 248	442	248	109616	5	plant	0.9136908
10	398 x 223	398	223	88754	5	plant	0.90853834
11	358 x 201	358	201	71958	4	plant	0.9059158
12	322 x 181	322	181	58282	4	plant	0.90615094
13	290 x 163	290	163	47270	3	plant	0.9168214
14	261 x 147	261	147	38367	3	plant	0.9003235
15	235 x 132	235	132	31020	3	plant	0.9165966
16	212 x 119	212	119	25228	2	plant	0.8883829
18	172 x 96	172	96	16512	2	plant	0.9071777

Figure 5.7 Label Detection for IMG_20161225_134039.jpg

The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *plant* word will display in their mind first. But the label detection detected *plant* description. Therefore, the minimum best image size display is in *191px x 107px* and plant description is fall on *1* seconds with the score *0.8982478.*

5.3 User Testing

Table 5.1 Click Icon Eidetic Search Testing Result

No	Event	Expected Result	Result
1.	User Click on icon.	Launch to the Main Menu Screen.	Pass

Table 5.2 SearchBar Testing Result

No	Event	Expected Result	Result
1.	User tap on the search bar and type keyword that user's wants.	Keyword being type as shown in screen.	Pass
2.	User click on search button.	Search image/images based on keywords typed.	Pass
3.	User tap on the advance filter search (Time) based on 1 hour ago, 1day ago,1 week ago, 1month ago and Click OK.	Image/Images will display based on the time value selected.	Pass

Table 5.3 Voice Recognition Search Testing Result

No	Event	Expected Result	Result
1.	User click on the mic icon to perform voice search recognition.	Search image/images based on keywords spoke.	Pass

Table 5.4 Start Camera Testing Result

No	Event	Expected Result	Result
1.	User click on the Start Camera button.	Navigate user to take photo module.	Pass
2.	User click on start to take photo.	Toast a radio button selection message will let user to choose how many capture duration (seconds) per photo taken.	Pass
3.	User selects value from radio button and click OK.	Camera will start capture based on the seconds selected. Eg. User chooses 1 sec, the auto-capture will perform capturing each photo per 1 second. A Toast message is display out telling user that camera has start.	Pass
4.	User click on Stop button.	Camera will be stop and a Toast Message is display out telling user camera has been stop.	

Table 5.5 Scan photos in local phone storage and transform it into database storage Testing Result

No	Event	Expected Result	Result
1.	User click on the Scan Photos button.	Photos will be scan and store all the description into database.	Pass

Table 5.6 User add Tag into Database Testing Result

No	Event	Expected Result	Result
1.	User taps on the photo and type any keyword within photo.	Tag added by user will stored into database.	Pass

Table 5.7 User delete Tag into Database Testing Result

No	Event	Expected Result	Result
1.	User taps on the photo and delete any keyword within photo.	Tag deleted by user will removed from database.	Pass

Table 5.8 User share photo to social media Testing Result

No	Event	Expected Result	Result
1.	User chooses photo and post in facebook.	Photo successfully posted in facebook.	Pass
2.	User chooses photo and post in whatapps.	Photo successfully posted in whatapps.	Pass
3.	User chooses photo and post in wechat.	Photo successfully posted in wechat.	Pass

5.2. Youtube Video Link

Search by Voice Recognition: https://youtu.be/1ZykXSLxO-U

Camera auto capturing: https://youtu.be/dZSfu0sGXmk

User search by text, add, delete tag: https://youtu.be/2WtamLWA1v0

User Scan photo information into database for data storage: https://youtu.be/oqPSsOMq6Pg

Share photo to social media: https://youtu.be/_emyeeMSN-g

Chapter 6 - Conclusion and Future Work

Even though nowadays, mobile based application able to provide similar searching images in local phone storage. What really makes this Eidetic Search mobile application unique from others are the objectives, it's satisfying the user demand by having tag photo management, which enables user to search, add, delete keyword in photos and retrieve it in a more convenience, shortest duration and more user-friendly way. For instance, it will make thorough analyses and filter out the probability score and other detail information within the photo using Google Cloud Vision API Label Detection and categorize it finely by listing the photos from descending order.

By having this improved system, Eidetic Search with significantly improved features such as: By using the image resizer function, the mobile application able to reduce the image size to minimize the best quality of the photos that are needed to send to the cloud, to implement the request Google Vision API detection in doing label indexing and abilities to help user to save the maximum data cellular network while uploading photos to perform Google Cloud Vision API services.

In future, the development of this system is trying to overcome the issues that faced by the current system, so that the user can reduce time consuming, while searching photo in an efficient and effective way, tag photo management to help user to organize the label within photos, like to add, delete and using the image resizer will reduce the maximum data network cellular consumption of the user when sending the request to the Google Cloud Vision API.

Appendix A

Installing Ionic

Ionic 2 apps are created and developed primarily through the Ionic command line utility (the "CLI"), and use Cordova to build and deploy as a native app. This means we need to install a few utilities to get developing.

Ionic CLI and Cordova

To create Ionic 2 projects, you'll need to install the latest version of the CLI and Cordova. Before you do that, you'll need a recent version of Node.js. Download the installer for Node.js 6 or greater and then proceed to install the Ionic CLI and Cordova for native app development:

```
$ npm install -g ionic cordova
```

You may need to add "sudo" in front of these commands to install the utilities globally
Once that's done, create your first Ionic app:

```
$ ionic start cutePuppyPics --v2
```

Omit –v2 if you'd like to use Ionic 1. To run your app, cd into the directory that was created and then run the ionic serve command to test your app right in the browser!

```
$ cd cutePuppyPics
$ ionic serve
```

Platform Guides

For those building native apps for iOS and Android (most of you!), each platform has certain features and installation requirements before you can get the most out of your Ionic and Cordova development.

For iOS developers, take a look at the Cordova iOS Platform Guide and follow the instructions to install or upgrade Xcode, and possibly register for a developer account to start building apps for iOS.

For Android developers, take a look at the Cordova Android Platform Guide and follow the instructions to install the SDK and/or Android Studio to start building apps for Android.

Installing Angular 2.0

```
$ npm install angular2
```

To upgrade, run:

```
$ [sudo] npm install npm@latest -g
```

Prerequisites

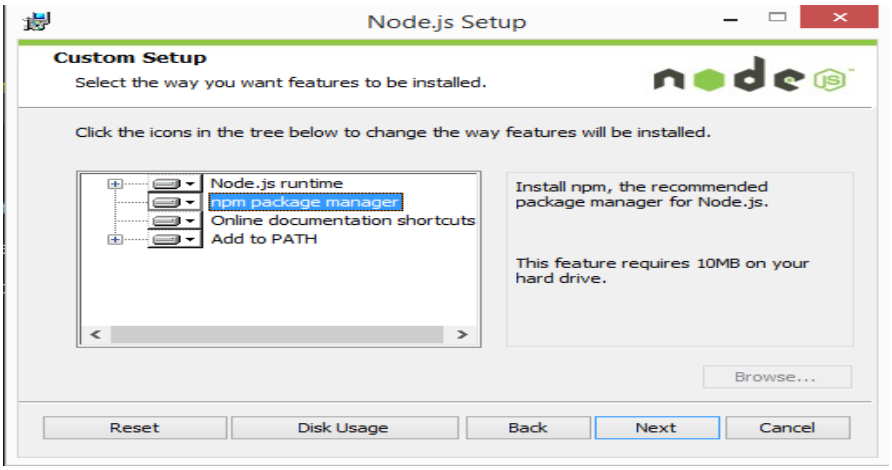
Node isn't a program that you simply launch like Word or Photoshop: you won't find it pinned to the taskbar or in your list of Apps. To use Node you must type command-line instructions, so you need to be comfortable with (or at least know how to start) a command-line tool like the Windows Command Prompt, PowerShell, Cygwin, or the Git shell (which is installed along with Github for Windows).

Installation Overview

Installing Node and NPM is pretty straightforward using the installer package available from the Node.js® web site.

Installation Steps

1. **Download the Windows installer from the Nodes.js® web site.**
2. **Run the installer** (the .msi file you downloaded in the previous step.)
3. **Follow the prompts in the installer** (Accept the license agreement, click the NEXT button a bunch of times and accept the default installation settings) installer.

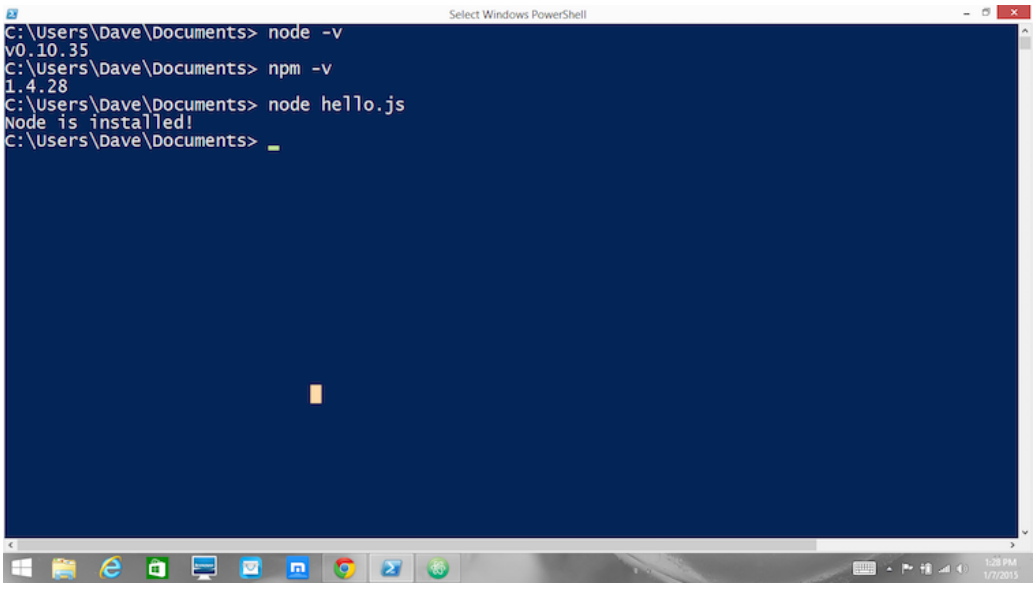


4. **Restart your computer**. You won't be able to run Node.js® until you restart your computer.

Test it!

Make sure you have Node and NPM installed by running simple commands to see what version of each is installed and to run a simple test program:

- **Test Node.** To see if Node is installed, open the Windows Command Prompt, Powershell or a similar command line tool, and type node -v. This should print a version number, so you'll see something like this v0.10.35.
- **Test NPM**. To see if NPM is installed, type npm -v in Terminal. This should print NPM's version number so you'll see something like this 1.4.28
- **Create a test file and run it**. A simple way to test that node.js works is to create a JavaScript file: name it hello.js, and just add the code console.log('Node is installed!');. To run the code simply open your command line program, navigate to the folder where you save the file and type node hello.js. This will start Node and run the code in the hello.js file. You should see the output Node is installed!



How to Update Node and NPM

New versions of Node and NPM come out frequently. To install the updates, just download the installer from the Nodejs.org site and run it again. The new version of Node and NPM will replace the older versions.

How to Uninstall Node and NPM

You uninstall Node.js and NPM the same as you would most Windows software:

1. Open the Windows Control Panel
2. Choose the “Programs and Features” option
3. Click the “Uninstall a program” option
4. Select Node.js, and click the Uninstall link.

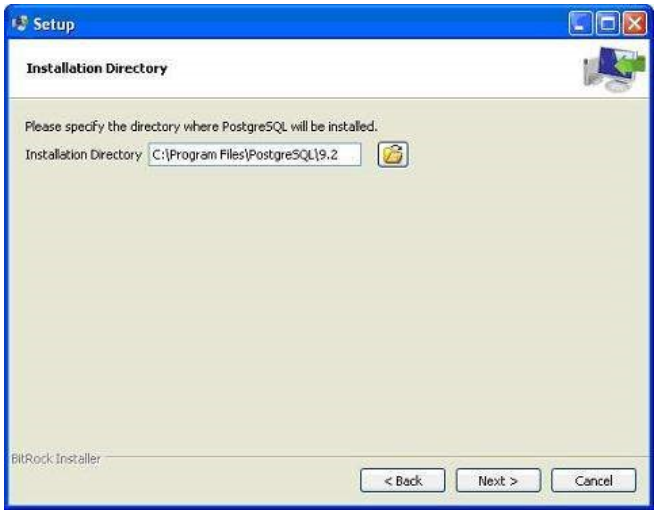
With Node.js and NPM installed you’ll soon be able to take advantage of the huge world of NPM modules that can help with a wide variety of tasks both on the web server and on your desktop (or laptop) machine. The NPM site lists all of the official Node packages making it easy to make the choice. Have fun and check out my current courses at Treehouse. And after, you’ve installed Node, check out the Node.js Basics course on Treehouse by my colleague, Andrew Chalkley.

Installing PostgreSQL on Windows

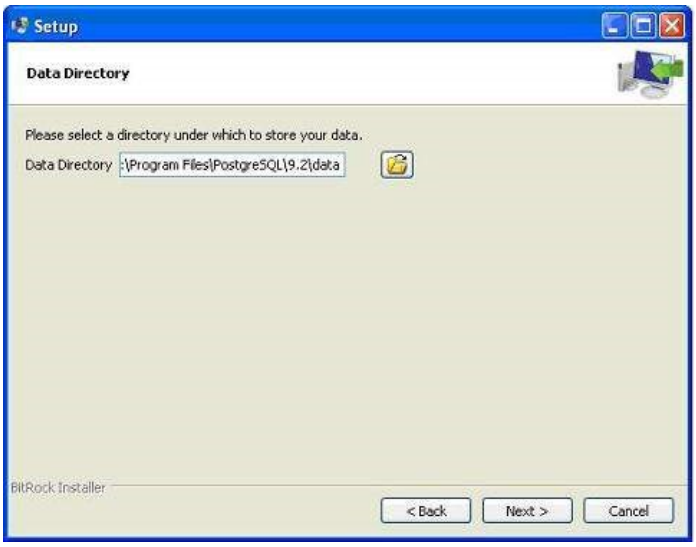
Follow the following steps to install PostgreSQL on your Windows machine. Make sure you have turned Third Party Antivirus off while installing.

- Pick the version number of PostgreSQL you want and, as exactly as possible, the platform you want from a EnterpriseDB

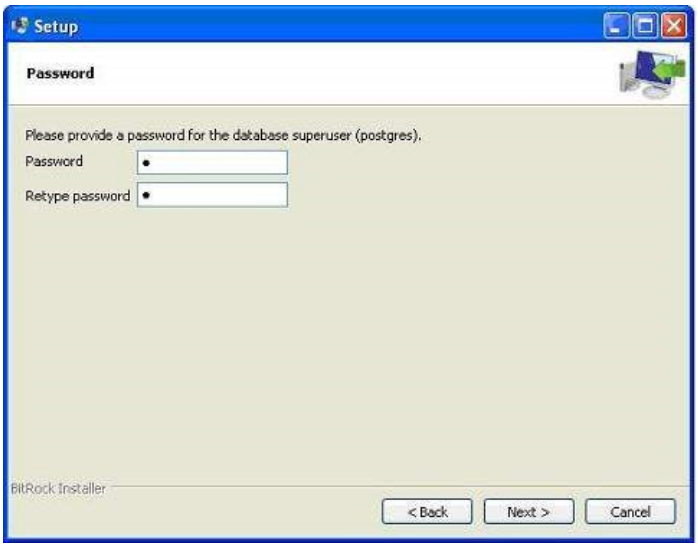
- I download postgresql-9.2.4-1-windows.exe for my Windows PC running in 32 bit mode, so lets run **postgresql-9.2.4-1-windows.exe** as administrator to install PostgreSQL. Select the location where you want to install it. By default it is installed within Program Files folder.



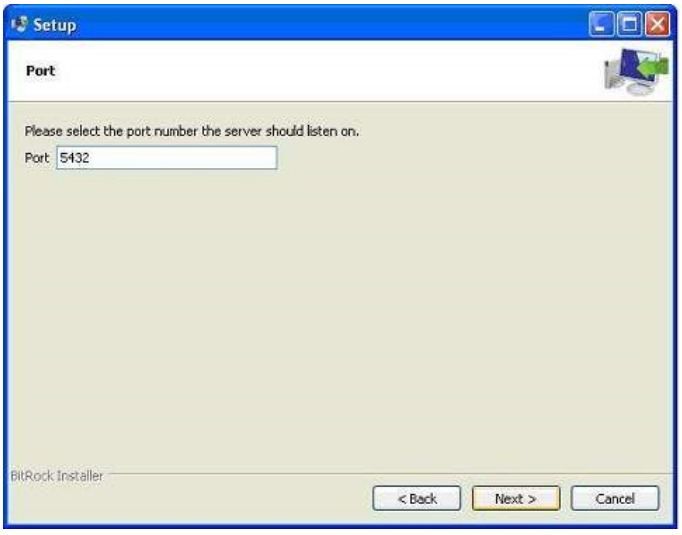
- The next step of the installation process would be to select the directory where data would be stored, by default it is stored under "data" directory



- The next step, setup asks for password, so you can use your favorite password

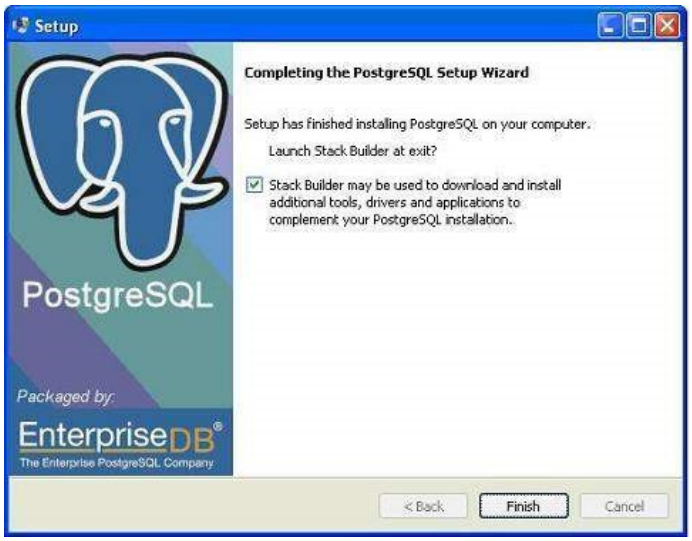


- The next step, keep the port as default



- The next step, when asked for "Locale", I have selected "English, United States".

- It takes a while to install PostgreSQL on your system. On completion of the installation process, you will get the following screen. Uncheck the checkbox and click on Finish button.



After the installation process is completed, you can access pgAdmin III, StackBuilder and PostgreSQL shell from your Program Menu under PostgreSQL 9.2.

https://cloud.google.com/vision/docs/common/auth

The generated JSON key will will be similar to the following sample JSON key:

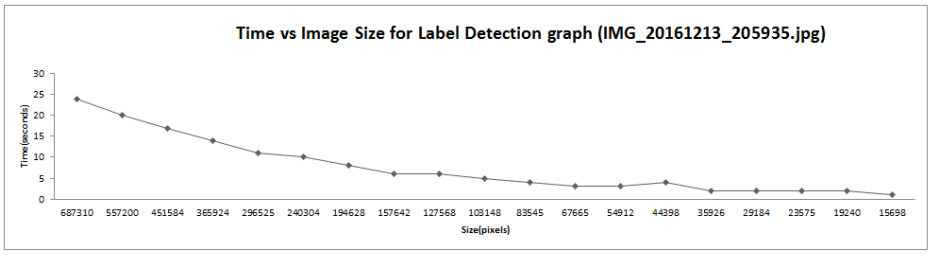
```
{

  "type": "service_account",

  "project_id": "project-id",

  "private_key_id": "some_number",

  "private_key": "-----BEGIN PRIVATE KEY-----\n....

  =\n-----END PRIVATE KEY-----\n",

  "client_email": "<api-name>api@project-
id.iam.gserviceaccount.com",

  "client_id": "...",

  "auth_uri": "https://accounts.google.com/o/oauth2/auth",

  "token_uri": "https://accounts.google.com/o/oauth2/token",
```

```
  "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",

  "client_x509_cert_url": "https://www.googleapis.com/...<api-
name>api%40project-id.iam.gserviceaccount.com"

}
```

Store this JSON file securely, as it contains your private key (and this file is the only copy of that key). You will need to refer to this service account key file within your code when you wish to send credentials to the Google Cloud Platform API.

Appendix B

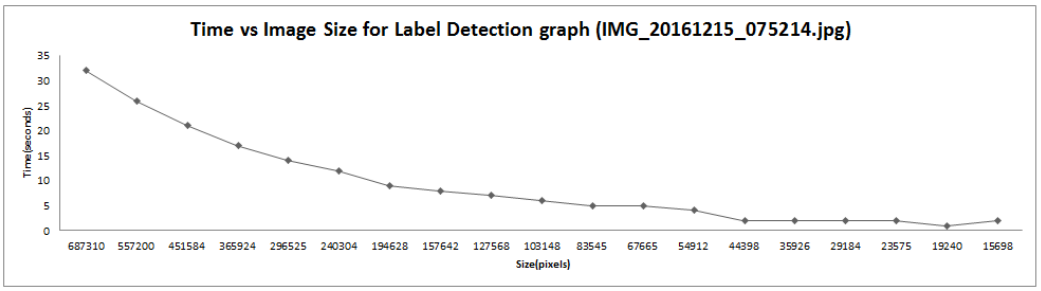
Image Result Testing Comparing Time Needed to do Label Detection using Google Cloud Vision API with Each Image Size reduced by 10 percent.



IMG_20161213_205935.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	622 x 1105	622	1105	687310	24	ice cream	0.91886324
2	560 x 995	560	995	557200	20	ice cream	0.9190913
3	504 x 896	504	896	451584	17	ice cream	0.9190102
4	454 x 806	454	806	365924	14	ice cream	0.91642684
5	409 x 725	409	725	296525	11	ice cream	0.92357814
6	368 x 653	368	653	240304	10	ice cream	0.92112947
7	331 x 588	331	588	194628	8	ice cream	0.925873
8	298 x 529	298	529	157642	6	ice cream	0.9222899
9	268 x 476	268	476	127568	6	ice cream	0.9263039
10	241 x 428	241	428	103148	5	ice cream	0.9202181
11	217 x 385	217	385	83545	4	ice cream	0.92604685
12	195 x 347	195	347	67665	3	ice cream	0.9235902
13	176 x 312	176	312	54912	3	ice cream	0.91228884
14	158 x 281	158	281	44398	4	ice cream	0.9125753
16	128 x 228	128	228	29184	2	food	0.9140951
17	115 x 205	115	205	23575	2	food	0.8810265
18	104 x 185	104	185	19240	2	food	0.88628435
19	94 x 167	94	167	15698	1	food	0.83071256

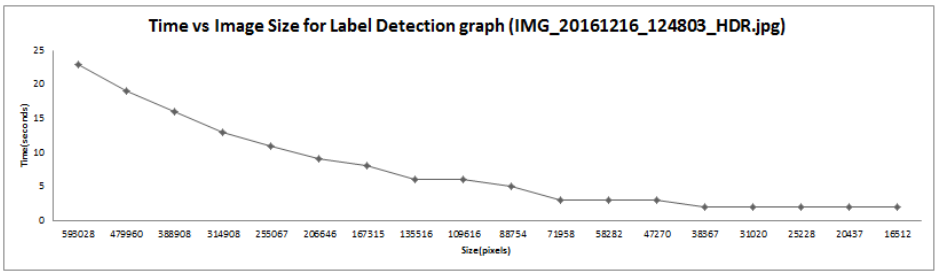
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *ice-cream* word will display in their mind first. Therefore, the minimum best image size display is in *142px x 253px* and *ice-cream* description is fall on *2* seconds with the score *0.89094764.*



IMG_20161215_075214.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	622 x 1105	622	1105	687310	32	face	0.9741372
2	560 x 995	560	995	557200	26	face	0.97353745
3	504 x 896	504	896	451584	21	face	0.9722727
4	454 x 806	454	806	365924	17	face	0.9743778
5	409 x 725	409	725	296525	14	face	0.97433877
6	368 x 653	368	653	240304	12	face	0.97312194
8	298 x 529	298	529	157642	8	face	0.97239935
9	268 x 476	268	476	127568	7	face	0.97287
10	241 x 428	241	428	103148	6	face	0.97409075
11	217 x 385	217	385	83545	5	face	0.97133756
12	195 x 347	195	347	67665	5	face	0.97162026
13	176 x 312	176	312	54912	4	face	0.9676548
14	158 x 281	158	281	44398	2	face	0.97070825
15	142 x 253	142	253	35926	2	face	0.964757
16	128 x 228	128	228	29184	2	face	0.9671888
17	115 x 205	115	205	23575	2	face	0.96656495
18	104 x 185	104	185	19240	1	face	0.9615646
19	94 x 167	94	167	15698	2	face	0.9552888

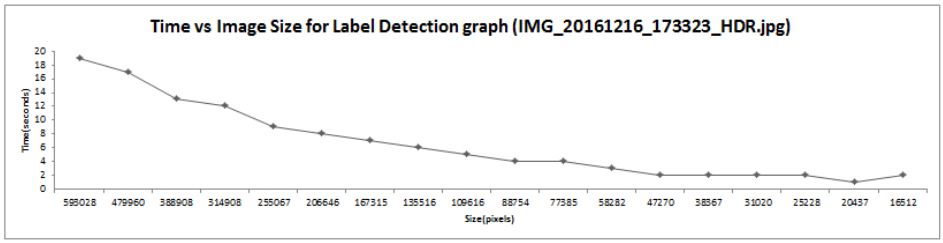
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *face* word will display in their mind first. Therefore, the minimum best image size display is in *104px x 185px* and *face* description is fall on *1* seconds with the score *0.9615646*.



IMG_20161216_124803_HDR.jpg

#	Image Size	Width	Height	Size(pixels)	Time(second	Description	Score
1	1026 x 578	1026	578	593028	23	food	0.95433414
2	923 x 520	923	520	479960	19	food	0.95492077
3	831 x 468	831	468	388908	16	food	0.95346576
4	748 x 421	748	421	314908	13	food	0.9534016
5	673 x 379	673	379	255067	11	food	0.95442516
6	606 x 341	606	341	206646	9	food	0.9524497
7	545 x 307	545	307	167315	8	food	0.95384985
8	491 x 276	491	276	135516	6	food	0.9531134
9	442 x 248	442	248	109616	6	food	0.95335585
10	398 x 223	398	223	88754	5	food	0.95010644
11	358 x 201	358	201	71958	3	food	0.951935
12	322 x 181	322	181	58282	3	food	0.9509136
13	290 x 163	290	163	47270	3	food	0.94855845
14	261 x 147	261	147	38367	2	food	0.948405
15	235 x 132	235	132	31020	2	food	0.94969094
16	212 x 119	212	119	25228	2	food	0.9529761
17	191 x 107	191	107	20437	2	food	0.95018196

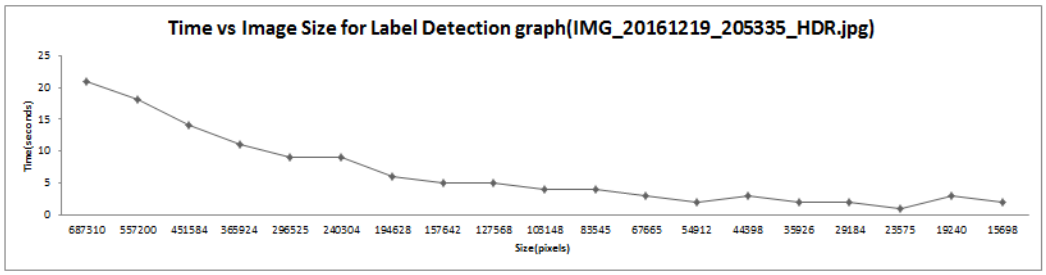
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *burger* or *fries* word will display in their mind first. But the label detection detected *food* description. Therefore, the minimum best image size display is in *172px x 96px* and food description is fall on *1* seconds with the score *0.95489347*.



IMG_20161216_173323_HDR.jpg

	Image Size	Width	Height	Size(pixels)	ime(Seconds	Description	Score
1	1026 x 578	1026	578	593028	19	airliner	0.9537918
2	923 x 520	923	520	479960	17	airliner	0.9559838
3	831 x 468	831	468	388908	13	airliner	0.95556074
4	748 x 421	748	421	314908	12	airliner	0.954089
5	673 x 379	673	379	255067	9	airline	0.9554799
6	606 x 341	606	341	206646	8	airline	0.9573363
7	545 x 307	545	307	167315	7	airline	0.9592908
8	491 x 276	491	276	135516	6	airline	0.96041864
9	442 x 248	442	248	109616	5	airline	0.95914996
10	398 x 223	398	223	88754	4	airliner	0.9519308
11	358 x 201	385	201	77385	4	airline	0.9553834
12	322 x 181	322	181	58282	3	airline	0.96010494
13	290 x 163	290	163	47270	2	airline	0.95772165
14	261 x 147	261	147	38367	2	airline	0.9610022
15	235 x 132	235	132	31020	2	airline	0.9620477
16	212 x 119	212	119	25228	2	airline	0.94250697
18	172 x 96	172	96	16512	2	airliner	0.89643025

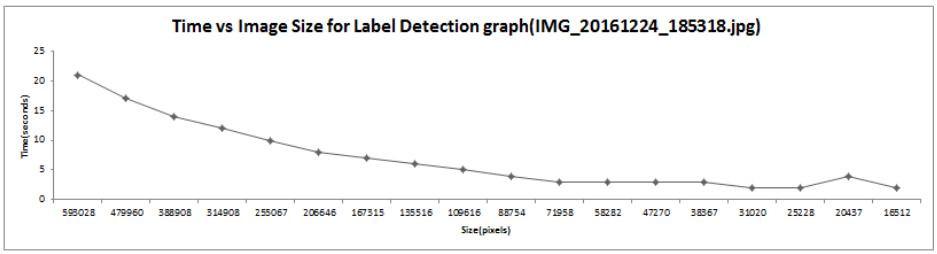
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *burger* or *fries* word will display in their mind first. But the label detection detected *food* description. Therefore, the minimum best image size display is in *172px x 96px* and airline description is fall on *1* seconds with the score *0.95489347.*



IMG_20161219_205335_HDR.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Secon	Description	Score
1	622 x 1105	622	1105	687310	21	food	0.88892305
2	560 x 995	560	995	557200	18	food	0.88160473
3	504 x 896	504	896	451584	14	food	0.8727476
4	454 x 806	454	806	365924	11	food	0.89285505
5	409 x 725	409	725	296525	9	food	0.8878762
6	368 x 653	368	653	240304	9	food	0.8890266
7	331 x 588	331	588	194628	6	food	0.8854653
8	298 x 529	298	529	157642	5	food	0.86070836
9	268 x 476	268	476	127568	5	food	0.88373584
10	241 x 428	241	428	103148	4	food	0.86409664
11	217 x 385	217	385	83545	4	food	0.83286375
12	195 x 347	195	347	67665	3	food	0.7961174
13	176 x 312	176	312	54912	2	food	0.8783301
14	158 x 281	158	281	44398	3	food	0.83608717
15	142 x 253	142	253	35926	2	food	0.8423535
17	115 x 205	115	205	23575	1	finger	0.8399555
18	104 x 185	104	185	19240	3	finger	0.7281927
19	94 x 167	94	167	15698	2	finger	0.7565564

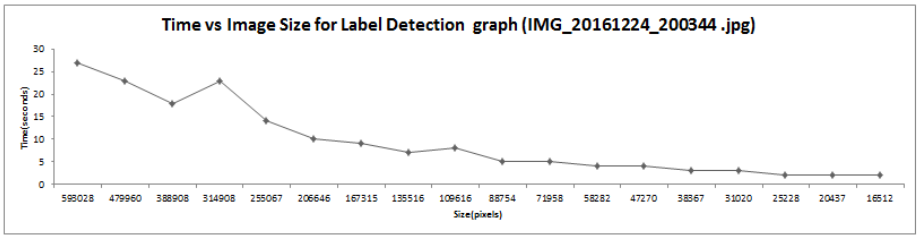
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *ice-cream stick* word will display in their mind first. But the label detection detected *food* description. Therefore, the minimum best image size display is in *128px x 228px* and food description is fall on *1* seconds with the score *0.854018*.



IMG_20161224_185318.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	1026 x 578	1026	578	593028	21	food	0.92457515
2	923 x 520	923	520	479960	17	food	0.93116707
3	831 x 468	831	468	388908	14	food	0.92875427
4	748 x 421	748	421	314908	12	food	0.9285074
5	673 x 379	673	379	255067	10	food	0.93089896
6	606 x 341	606	341	206646	8	food	0.9252605
7	545 x 307	545	307	167315	7	food	0.9301973
8	491 x 276	491	276	135516	6	food	0.9231103
9	442 x 248	442	248	109616	5	food	0.9313149
10	398 x 223	398	223	88754	4	food	0.940264
11	358 x 201	358	201	71958	3	food	0.93196356
12	322 x 181	322	181	58282	3	food	0.92136955
13	290 x 163	290	163	47270	3	food	0.91276383
14	261 x 147	261	147	38367	3	food	0.93130475
15	235 x 132	235	132	31020	2	food	0.9266505
16	212 x 119	212	119	25228	2	food	0.91329974
17	191 x 107	191	107	20437	4	food	0.93881655

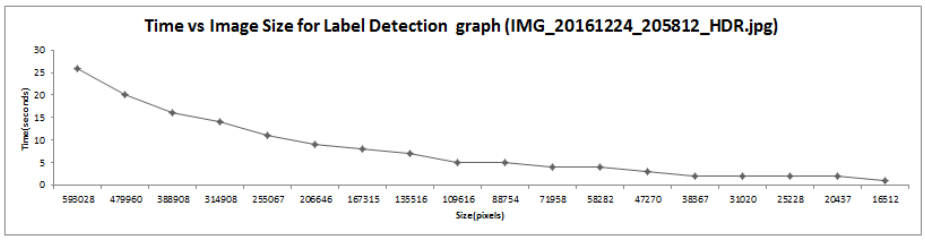
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *prawn* word will display in their mind first. But the label detection detected *food* description. Therefore, the minimum best image size display is in *172px x 96px* and food description is fall on *2* seconds with the score *0.9413979*.



IMG_20161224_200344.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	1026 x 578	1026	578	593028	27	metropolitan area	0.89671844
2	923 x 520	923	520	479960	23	metropolitan area	0.90177286
3	831 x 468	831	468	388908	18	metropolitan area	0.896159
4	748 x 421	748	421	314908	23	metropolitan area	0.93477964
5	673 x 379	673	379	255067	14	metropolitan area	0.87218434
6	606 x 341	606	341	206646	10	metropolitan area	0.86588675
7	545 x 307	545	307	167315	9	night	0.8482648
8	491 x 276	491	276	135516	7	night	0.8493188
9	442 x 248	442	248	109616	8	landmark	0.8447707
10	398 x 223	398	223	88754	5	christmas lights	0.83731574
11	358 x 201	358	201	71958	5	christmas lights	0.842682
12	322 x 181	322	181	58282	4	christmas lights	0.8620049
13	290 x 163	290	163	47270	4	christmas lights	0.8618988
14	261 x 147	261	147	38367	3	christmas lights	0.86780363
15	235 x 132	235	132	31020	3	christmas lights	0.8747766
16	212 x 119	212	119	25228	2	christmas lights	0.8754165
17	191 x 107	191	107	20437	2	christmas lights	0.8894585

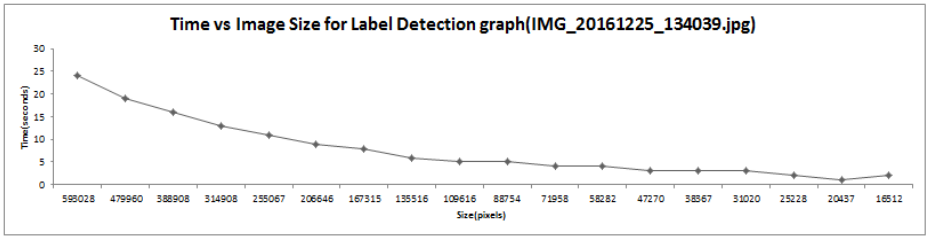
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *Christmas tree* word will display in their mind first. But the label detection detected *Christmas lights* description. Therefore, the minimum best image size display is in *172px x 96px* and Christmas lights description is fall on *2* seconds with the score *0.88195306.*



IMG_20161224_205812_HDR.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Secon	Description	Score
1	1026 x 578	1026	578	593028	26	night	0.8696515
2	923 x 520	923	520	479960	20	night	0.87143135
3	831 x 468	831	468	388908	16	night	0.8626419
4	748 x 421	748	421	314908	14	night	0.8591858
5	673 x 379	673	379	255067	11	night	0.865366
6	606 x 341	606	341	206646	9	night	0.8685078
7	545 x 307	545	307	167315	8	night	0.8579482
8	491 x 276	491	276	135516	7	landmark	0.83433723
9	442 x 248	442	248	109616	5	night	0.8160931
10	398 x 223	398	223	88754	5	night	0.7538067
11	358 x 201	358	201	71958	4	plaza	0.5861907
12	322 x 181	322	181	58282	4	christmas lights	0.53189427
13	290 x 163	290	163	47270	3	christmas decoration	0.5657912
14	261 x 147	261	147	38367	2	christmas lights	0.5136988
15	235 x 132	235	132	31020	2	christmas decoration	0.55361855
16	212 x 119	212	119	25228	2	christmas lights	0.5851601
17	191 x 107	191	107	20437	2	christmas decoration	0.7358428

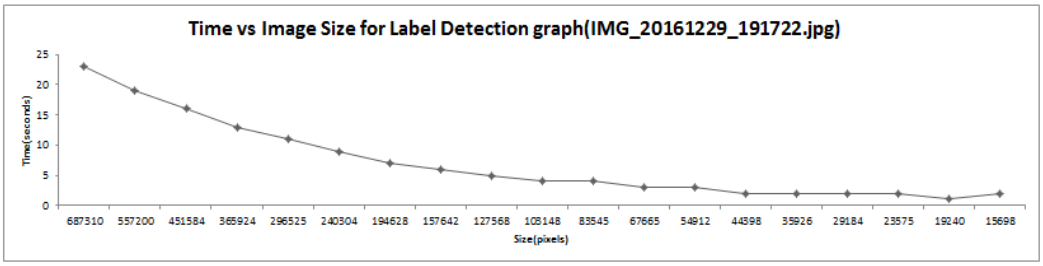
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *heritage building* word will display in their mind first. But the label detection detected *Christmas decoration* description. Therefore, the minimum best image size display is in *172px x 96px* and Christmas decoration description is fall on *1* seconds with the score *0.7662213.*



IMG_20161225_134039.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	1026 x 578	1026	578	593028	24	green	0.93538237
2	923 x 520	923	520	479960	19	green	0.9304343
3	831 x 468	831	468	388908	16	green	0.9343031
4	748 x 421	748	421	314908	13	green	0.92323303
5	673 x 379	673	379	255067	11	green	0.9240687
6	606 x 341	606	341	206646	9	green	0.92584044
7	545 x 307	545	307	167315	8	green	0.91922164
8	491 x 276	491	276	135516	6	plant	0.9059518
9	442 x 248	442	248	109616	5	plant	0.9136908
10	398 x 223	398	223	88754	5	plant	0.90853834
11	358 x 201	358	201	71958	4	plant	0.9059158
12	322 x 181	322	181	58282	4	plant	0.90615094
13	290 x 163	290	163	47270	3	plant	0.9168214
14	261 x 147	261	147	38367	3	plant	0.9003235
15	235 x 132	235	132	31020	3	plant	0.9165966
16	212 x 119	212	119	25228	2	plant	0.8883829
18	172 x 96	172	96	16512	2	plant	0.9071777

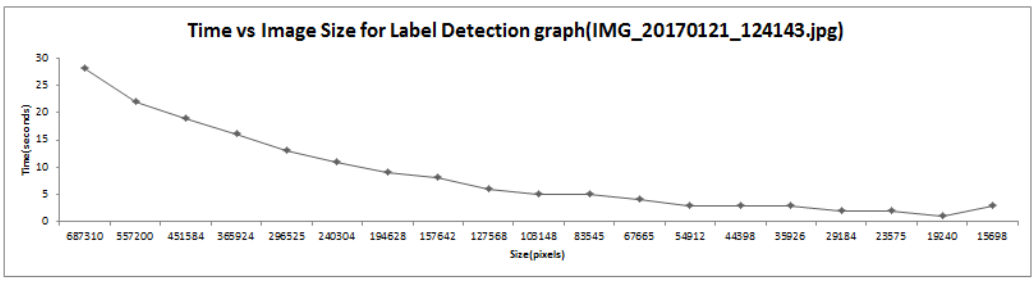
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *plant* word will display in their mind first. But the label detection detected *plant* description. Therefore, the minimum best image size display is in *191px x 107px* and plant description is fall on *1* seconds with the score *0.8982478.*



IMG_20161229_191722.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	622 x 1105	622	1105	687310	23	blue	0.9273284
2	560 x 995	560	995	557200	19	blue	0.9272723
3	504 x 896	504	896	451584	16	blue	0.9274491
4	454 x 806	454	806	365924	13	blue	0.9254833
5	409 x 725	409	725	296525	11	blue	0.9237771
6	368 x 653	368	653	240304	9	blue	0.9137443
7	331 x 588	331	588	194628	7	blue	0.90717274
8	298 x 529	298	529	157642	6	swimming pool	0.7064741
9	268 x 476	268	476	127568	5	blue	0.9033597
10	241 x 428	241	428	103148	4	swimming pool	0.7470694
11	217 x 385	217	385	83545	4	swimming pool	0.79031557
12	195 x 347	195	347	67665	3	swimming pool	0.75343835
13	176 x 312	176	312	54912	3	swimming pool	0.826866
14	158 x 281	158	281	44398	2	swimming pool	0.8075441
15	142 x 253	142	253	35926	2	swimming pool	0.801422
16	128 x 228	128	228	29184	2	boat	0.9432393
17	115 x 205	115	205	23575	2	boat	0.9443611
19	94 x 167	94	167	15698	2	boat	0.93280864

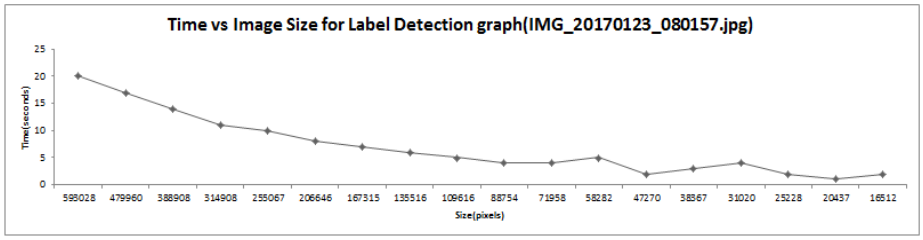
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the coconut water word will display in their mind first. But the label detection detected *boat* description. Therefore, the minimum best image size display is in *104px x 185px* and boat description is fall on *1* seconds with the score *0.8807535*.



IMG_20170121_124143.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	622 x 1105	622	1105	687310	28	nutcracker	0.95652527
2	560 x 995	560	995	557200	22	nutcracker	0.9563998
3	504 x 896	504	896	451584	19	nutcracker	0.95664996
4	454 x 806	454	806	365924	16	nutcracker	0.9565131
5	409 x 725	409	725	296525	13	nutcracker	0.9560024
6	368 x 653	368	653	240304	11	nutcracker	0.95675254
7	331 x 588	331	588	194628	9	nutcracker	0.95671976
8	298 x 529	298	529	157642	8	nutcracker	0.9563755
9	268 x 476	268	476	127568	6	nutcracker	0.9567421
10	241 x 428	241	428	103148	5	nutcracker	0.95730865
11	217 x 385	217	385	83545	5	nutcracker	0.9573308
12	195 x 347	195	347	67665	4	nutcracker	0.9569386
13	176 x 312	176	312	54912	3	nutcracker	0.95726764
14	158 x 281	158	281	44398	3	nutcracker	0.95693564
15	142 x 253	142	253	35926	3	nutcracker	0.95669985
16	128 x 228	128	228	29184	2	nutcracker	0.9570469
17	115 x 205	115	205	23575	2	nutcracker	0.94715446
19	94 x 167	94	167	15698	3	nutcracker	0.94463044

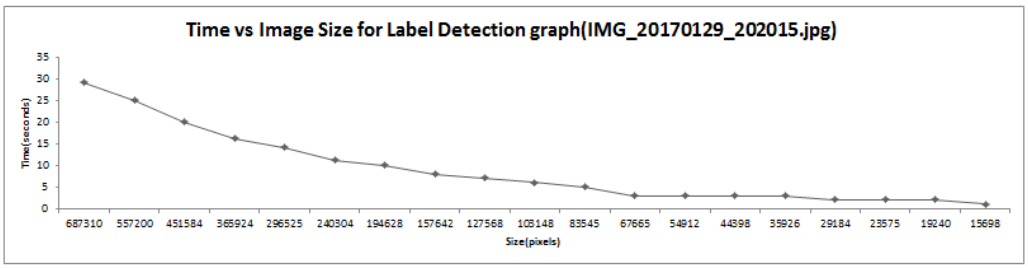
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the coconut water word will display in their mind first. But the label detection detected *boat* description. Therefore, the minimum best image size display is in *104px x 185px* and nutcracker description is fall on *1* seconds with the score *0.8807535.*



IMG_20170123_080157.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Secon	Description	Score
1	1026 x 578	1026	578	593028	20	green	0.89207745
2	923 x 520	923	520	479960	17	green	0.8951716
3	831 x 468	831	468	388908	14	green	0.8846459
4	748 x 421	748	421	314908	11	green	0.8933938
5	673 x 379	673	379	255067	10	green	0.8789531
6	606 x 341	606	341	206646	8	green	0.8716971
7	545 x 307	545	307	167315	7	green	0.86544997
8	491 x 276	491	276	135516	6	green	0.86639994
9	442 x 248	442	248	109616	5	land lot	0.8278995
10	398 x 223	398	223	88754	4	land lot	0.80409503
11	358 x 201	358	201	71958	4	land lot	0.8486352
12	322 x 181	322	181	58282	5	land lot	0.83221024
13	290 x 163	290	163	47270	2	land lot	0.84888136
14	261 x 147	261	147	38367	3	property	0.86616254
15	235 x 132	235	132	31020	4	property	0.867732
17	191 x 107	191	107	20437	1	property	0.8782732
18	172 x 96	172	96	16512	2	agriculture	0.84681594

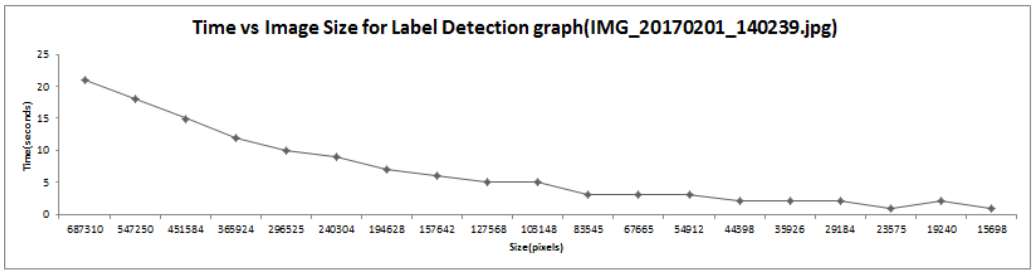
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the path word will display in their mind first. But the label detection detected *property* description. Therefore, the minimum best image size display is in *212px x 119px* and property description is fall on *2* seconds with the score *0.855566723.*



IMG_20170129_202015.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Secon	Description	Score
1	622 x 1105	622	1105	687310	29	alcoholic beverage	0.8801463
2	560 x 995	560	995	557200	25	alcoholic beverage	0.8757975
3	504 x 896	504	896	451584	20	alcoholic beverage	0.8746322
4	454 x 806	454	806	365924	16	alcoholic beverage	0.8752315
5	409 x 725	409	725	296525	14	alcoholic beverage	0.884237
6	368 x 653	368	653	240304	11	alcoholic beverage	0.8795511
7	331 x 588	331	588	194628	10	alcoholic beverage	0.8753553
8	298 x 529	298	529	157642	8	alcoholic beverage	0.87999374
9	268 x 476	268	476	127568	7	alcoholic beverage	0.8726357
10	241 x 428	241	428	103148	6	alcoholic beverage	0.88601047
11	217 x 385	217	385	83545	5	alcoholic beverage	0.884417
12	195 x 347	195	347	67665	3	alcoholic beverage	0.89334023
13	176 x 312	176	312	54912	3	alcoholic beverage	0.8762752
14	158 x 281	158	281	44398	3	alcoholic beverage	0.88955617
15	142 x 253	142	253	35926	3	alcoholic beverage	0.8638363
16	128 x 228	128	228	29184	2	alcoholic beverage	0.89796525
17	115 x 205	115	205	23575	2	alcoholic beverage	0.8849772
18	104 x 185	104	185	19240	2	alcoholic beverage	0.8905134

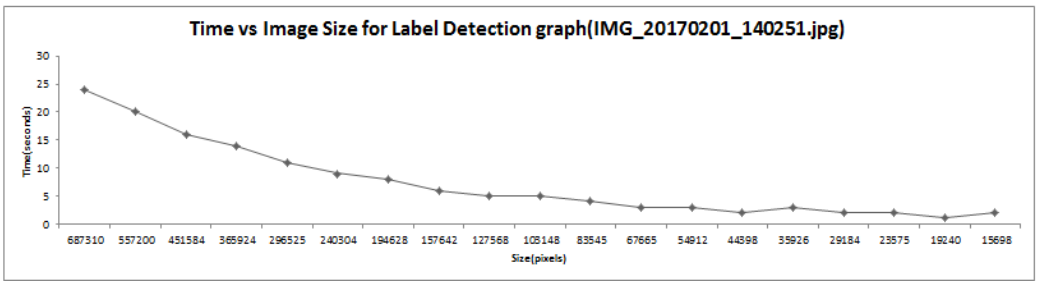
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the wine word will display in their mind first. But the label detection detected *alcoholic beverage* description. Therefore, the minimum best image size display is in *94px x 167px* and alcoholic beverage description is fall on *1* seconds with the score *0.9248843*.



IMG_20170201_140239.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Seconds)	Description	Score
1	622 x 1105	622	1105	687310	21	vehicle	0.9107485
2	560 x 995	550	995	547250	18	vehicle	0.91822326
3	504 x 896	504	896	451584	15	vehicle	0.9272622
4	454 x 806	454	806	365924	12	car	0.9306417
5	409 x 725	409	725	296525	10	vehicle	0.9329963
6	368 x 653	368	653	240304	9	vehicle	0.93021077
7	331 x 588	331	588	194628	7	vehicle	0.91415393
8	298 x 529	298	529	157642	6	car	0.9146473
9	268 x 476	268	476	127568	5	vehicle	0.90814143
10	241 x 428	241	428	103148	5	vehicle	0.8983074
11	217 x 385	217	385	83545	3	vehicle	0.9177212
12	195 x 347	195	347	67665	3	transport	0.88434696
13	176 x 312	176	312	54912	3	vehicle	0.9016654
14	158 x 281	158	281	44398	2	vehicle	0.91133726
15	142 x 253	142	253	35926	2	vehicle	0.9278291
16	128 x 228	128	228	29184	2	land vehicle	0.9108143
17	115 x 205	115	205	23575	1	vehicle	0.94314647
18	104 x 185	104	185	19240	2	vehicle	0.8904039

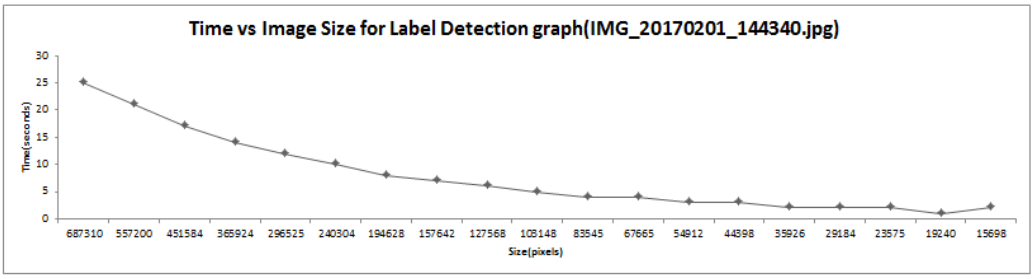
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *parking lot* word will display in their mind first. But the label detection detected *vehicle* description. Therefore, the minimum best image size display is in *94px x 167px* and vehicle description is fall on *1* seconds with the score *0.92935026.*



IMG_20170201_140251.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Secon	Description	Score
1	622 x 1105	622	1105	687310	24	car	0.93856543
2	560 x 995	560	995	557200	20	car	0.940558
3	504 x 896	504	896	451584	16	car	0.92905676
4	454 x 806	454	806	365924	14	car	0.9482939
5	409 x 725	409	725	296525	11	car	0.9420969
6	368 x 653	368	653	240304	9	car	0.93871737
7	331 x 588	331	588	194628	8	car	0.92043376
8	298 x 529	298	529	157642	6	car	0.94865906
9	268 x 476	268	476	127568	5	car	0.94906765
10	241 x 428	241	428	103148	5	car	0.9392613
11	217 x 385	217	385	83545	4	car	0.94470835
12	195 x 347	195	347	67665	3	car	0.91515756
13	176 x 312	176	312	54912	3	car	0.9176358
14	158 x 281	158	281	44398	2	car	0.9169187
15	142 x 253	142	253	35926	3	car	0.86892074
16	128 x 228	128	228	29184	2	structure	0.88276213
17	115 x 205	115	205	23575	2	mode of transport	0.8423645
19	94 x 167	94	167	15698	2	mode of transport	0.8423645

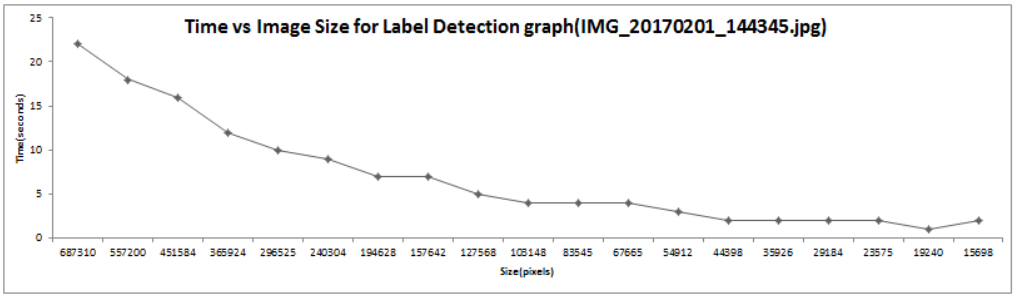
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *car* word will display in their mind first. But the label detection detected *mode of transport* description. Therefore, the minimum best image size display is in *104px x 185px* and mode of transport description is fall on *1* seconds with the score *0.8423645*.



IMG_20170201_144340.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Secon	Description	Score
1	622 x 1105	622	1105	687310	25	food	0.94283617
2	560 x 995	560	995	557200	21	food	0.9437701
3	504 x 896	504	896	451584	17	food	0.9441151
4	454 x 806	454	806	365924	14	food	0.94338
5	409 x 725	409	725	296525	12	food	0.9431473
6	368 x 653	368	653	240304	10	food	0.9436793
7	331 x 588	331	588	194628	8	food	0.9424228
8	298 x 529	298	529	157642	7	food	0.9429462
9	268 x 476	268	476	127568	6	food	0.940852
10	241 x 428	241	428	103148	5	food	0.9440684
11	217 x 385	217	385	83545	4	dish	0.939374
12	195 x 347	195	347	67665	4	dish	0.94269127
13	176 x 312	176	312	54912	3	dish	0.93915415
14	158 x 281	158	281	44398	3	dish	0.94309103
15	142 x 253	142	253	35926	2	dish	0.93994445
16	128 x 228	128	228	29184	2	dish	0.93920165
17	115 x 205	115	205	23575	2	dish	0.94184345
19	94 x 167	94	167	15698	2	dish	0.9366719

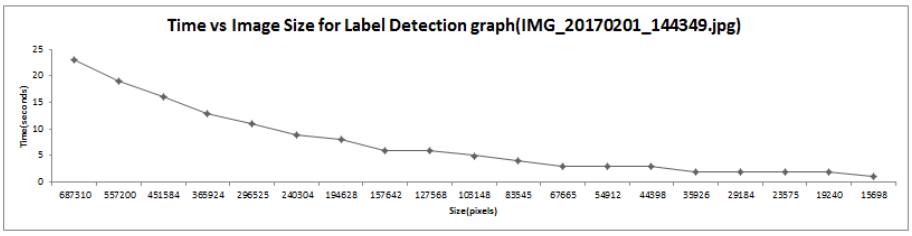
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *chicken wings* word will display in their mind first. But the label detection detected *dish* description. Therefore, the minimum best image size display is in *104px x 185px* and dish description is fall on *1* seconds with the score *0.9430063*.



IMG_20170201_144345.jpg

	Image Size	Width	Height	Size(pixels)	ime(Seconds	Description	Score
1	622 x 1105	622	1105	687310	22	dish	0.93452007
2	560 x 995	560	995	557200	18	dish	0.9363059
3	504 x 896	504	896	451584	16	dish	0.9356379
4	454 x 806	454	806	365924	12	dish	0.93380785
5	409 x 725	409	725	296525	10	dish	0.9329634
6	368 x 653	368	653	240304	9	dish	0.93280756
7	331 x 588	331	588	194628	7	dish	0.93506795
8	298 x 529	298	529	157642	7	dish	0.93960977
9	268 x 476	268	476	127568	5	dish	0.938349
10	241 x 428	241	428	103148	4	dish	0.93809134
11	217 x 385	217	385	83545	4	dish	0.93859017
12	195 x 347	195	347	67665	4	dish	0.9388002
13	176 x 312	176	312	54912	3	dish	0.94087267
14	158 x 281	158	281	44398	2	dish	0.94508916
15	142 x 253	142	253	35926	2	dish	0.9420198
16	128 x 228	128	228	29184	2	food	0.94732
17	115 x 205	115	205	23575	2	food	0.95497346
19	94 x 167	94	167	15698	2	food	0.95842373

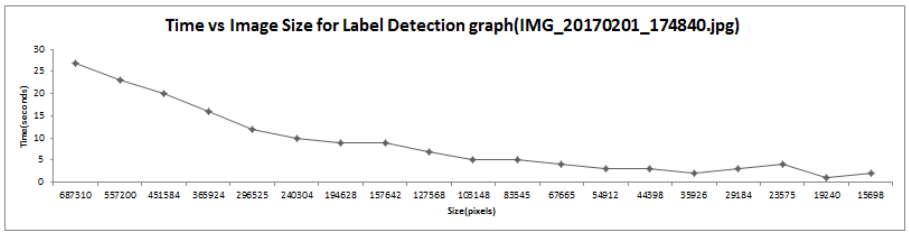
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the Tomato sauces *macaroni* word will display in their mind first. But the label detection detected *dish* description. Therefore, the minimum best image size display is in *104px x 185px* and dish description is fall on *1* seconds with the score *0.9478677.*



IMG_20170201_144349.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Secon	Description	Score
1	622 x 1105	622	1105	687310	23	food	0.95082676
2	560 x 995	560	995	557200	19	food	0.94997424
3	504 x 896	504	896	451584	16	food	0.9502429
4	454 x 806	454	806	365924	13	food	0.949884
5	409 x 725	409	725	296525	11	food	0.9512668
6	368 x 653	368	653	240304	9	food	0.9488542
7	331 x 588	331	588	194628	8	food	0.9478599
8	298 x 529	298	529	157642	6	food	0.9509123
9	268 x 476	268	476	127568	6	food	0.94518024
10	241 x 428	241	428	103148	5	food	0.94387686
11	217 x 385	217	385	83545	4	food	0.9454342
12	195 x 347	195	347	67665	3	food	0.939625
13	176 x 312	176	312	54912	3	food	0.9383662
14	158 x 281	158	281	44398	3	dish	0.94176495
15	142 x 253	142	253	35926	2	dish	0.94319385
16	128 x 228	128	228	29184	2	dish	0.94482833
17	115 x 205	115	205	23575	2	food	0.9472208
18	104 x 185	104	185	19240	2	dish	0.9412026

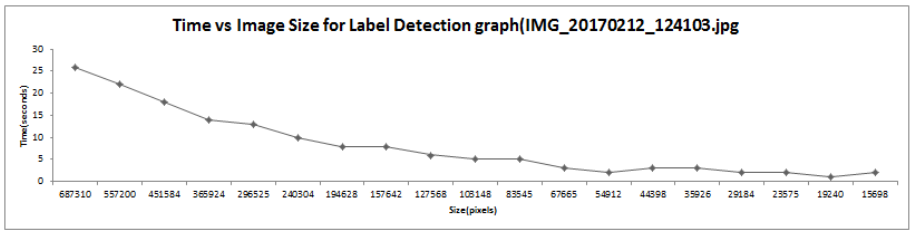
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the nugget fries word will display in their mind first. But the label detection detected food description. Therefore, the minimum best image size display is in *94px x 167px* and food description is fall on *1* seconds with the score *0.94350713*.



IMG_20170201_174840.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Secon	Description	Score
1	622 x 1105	622	1105	687310	27	clothing	0.92047185
2	560 x 995	560	995	557200	23	clothing	0.9230245
3	504 x 896	504	896	451584	20	clothing	0.92332244
4	454 x 806	454	806	365924	16	clothing	0.92346036
5	409 x 725	409	725	296525	12	clothing	0.9224558
6	368 x 653	368	653	240304	10	clothing	0.91816425
7	331 x 588	331	588	194628	9	clothing	0.91538924
8	298 x 529	298	529	157642	9	clothing	0.9177928
9	268 x 476	268	476	127568	7	clothing	0.9118995
10	241 x 428	241	428	103148	5	clothing	0.913961
11	217 x 385	217	385	83545	5	clothing	0.9135309
12	195 x 347	195	347	67665	4	clothing	0.90565884
13	176 x 312	176	312	54912	3	clothing	0.90607363
14	158 x 281	158	281	44398	3	clothing	0.9062235
15	142 x 253	142	253	35926	2	person	0.8977419
16	128 x 228	128	228	29184	3	costume	0.51765466
17	115 x 205	115	205	23575	4	clothing	0.90959805
18	104 x 185	104	185	19240	1	costume	0.5237505

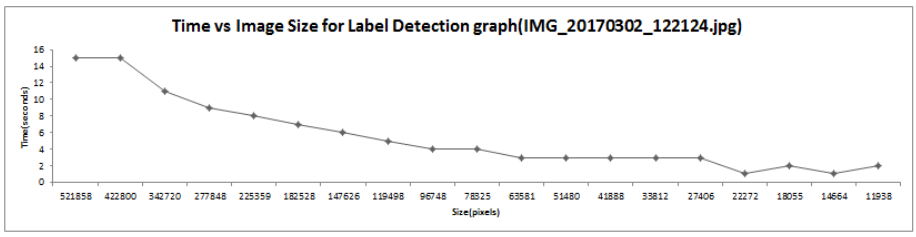
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *girls* word will display in their mind first. But the label detection detected *clothing* description. Therefore, the minimum best image size display is in *94px x 165px* and clothing description is fall on *2* seconds with the score *0.9034073*.

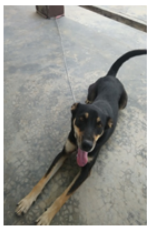


IMG_20170212_124103.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	622 x 1105	622	1105	687310	26	color	0.9688637
2	560 x 995	560	995	557200	22	color	0.9688637
3	504 x 896	504	896	451584	18	color	0.9688637
4	454 x 806	454	806	365924	14	color	0.9688637
5	409 x 725	409	725	296525	13	red	0.8882006
6	368 x 653	368	653	240304	10	color	0.9688637
7	331 x 588	331	588	194628	8	room	0.8423869
8	298 x 529	298	529	157642	8	color	0.9688637
9	268 x 476	268	476	127568	6	room	0.8124269
10	241 x 428	241	428	103148	5	room	0.8212213
11	217 x 385	217	385	83545	5	property	0.8707445
12	195 x 347	195	347	67665	3	property	0.8736465
13	176 x 312	176	312	54912	2	property	0.869535
14	158 x 281	158	281	44398	3	property	0.8713434
15	142 x 253	142	253	35926	3	room	0.7855317
16	128 x 228	128	228	29184	2	product	0.71492827
17	115 x 205	115	205	23575	2	room	0.83385515
19	94 x 167	94	167	15698	2	room	0.82763207

The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *front door* word will display in their mind first. But the label detection detected *art* description. Therefore, the minimum best image size display is in *104px x 185px* and art description is fall on *1* seconds with the score *0.76519084.*

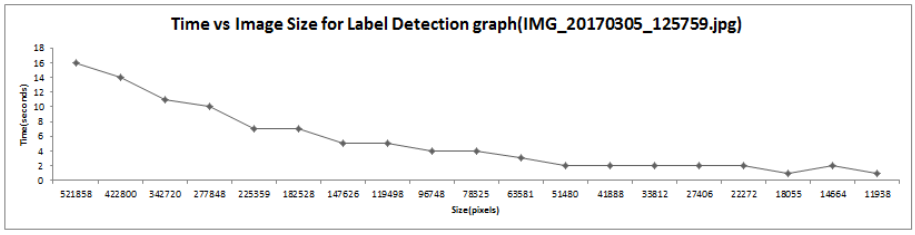




IMG_20170302_122124.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	622 x 839	622	839	521858	15	dog	0.989354
2	560 x 755	560	755	422800	15	dog	0.98825586
3	504 x 680	504	680	342720	11	dog	0.9899717
4	454 x 612	454	612	277848	9	dog	0.9892042
5	409 x 551	409	551	225359	8	dog	0.98970616
6	368 x 496	368	496	182528	7	dog	0.99126536
7	331 x 446	331	446	147626	6	dog	0.9885473
8	298 x 401	298	401	119498	5	dog	0.99126375
9	268 x 361	268	361	96748	4	dog	0.99053264
10	241 x 325	241	325	78325	4	dog	0.99217093
11	217 x 293	217	293	63581	3	dog	0.9904518
12	195 x 264	195	264	51480	3	dog	0.9928805
13	176 x 238	176	238	41888	3	dog	0.9913857
14	158 x 214	158	214	33812	3	dog	0.9930385
15	142 x 193	142	193	27406	3	dog	0.9937246
16	128 x 174	128	174	22272	1	dog	0.99050224
17	115 x 157	115	157	18055	2	dog	0.9910014
19	94 x 127	94	127	11938	2	dog	0.9829607

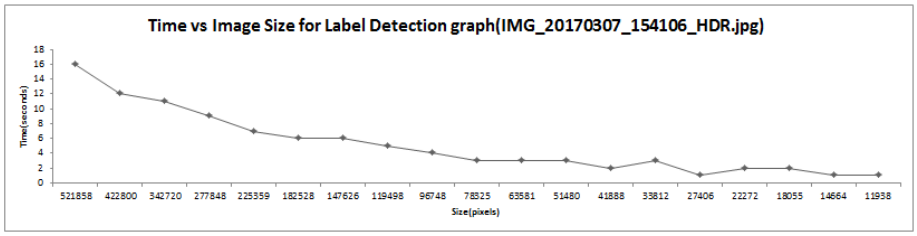
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *dog* word will display in their mind first. But the label detection detected *dog* description. Therefore, the minimum best image size display is in *104px x 141px* and dog description is fall on *1* seconds with the score *0.9913005*.



IMG_20170305_125759.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	622 x 839	622	839	521858	16	clothing	0.9083088
2	560 x 755	560	755	422800	14	clothing	0.9081423
3	504 x 680	504	680	342720	11	clothing	0.9045825
4	454 x 612	454	612	277848	10	clothing	0.9057528
5	409 x 551	409	551	225359	7	clothing	0.9089773
6	368 x 496	368	496	182528	7	clothing	0.9095871
7	331 x 446	331	446	147626	5	clothing	0.9096041
8	298 x 401	298	401	119498	5	clothing	0.91848373
9	268 x 361	268	361	96748	4	clothing	0.9088917
10	241 x 325	241	325	78325	4	clothing	0.9069106
11	217 x 293	217	293	63581	3	clothing	0.91545236
12	195 x 264	195	264	51480	2	clothing	0.9060358
13	176 x 238	176	238	41888	2	clothing	0.76111525
14	158 x 214	158	214	33812	2	clothing	0.90542686
15	142 x 193	142	193	27406	2	clothing	0.9162202
16	128 x 174	128	174	22272	2	clothing	0.9133883
17	115 x 157	115	157	18055	1	clothing	0.9194575
18	104 x 141	104	141	14664	2	clothing	0.9174856

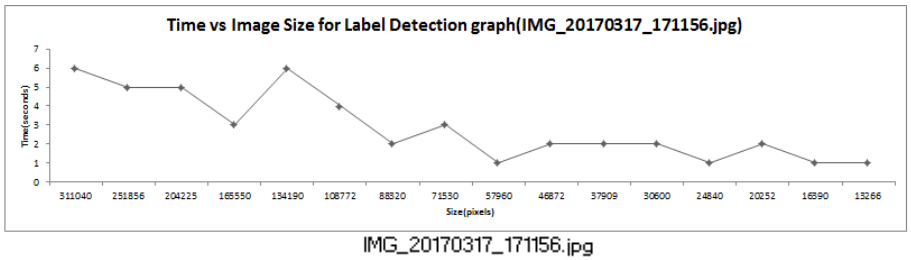
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *little boy* word will display in their mind first. But the label detection detected *clothing* description. Therefore, the minimum best image size display is in *104px x 185px* and clothing description is fall on *1* seconds with the score *0.9145716.*



IMG_20170307_154106_HDR.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Secon	Description	Score
1	622 x 839	622	839	521858	16	red	0.9088064
2	560 x 755	560	755	422800	12	red	0.8938789
3	504 x 680	504	680	342720	11	red	0.9025592
4	454 x 612	454	612	277848	9	red	0.9049086
5	409 x 551	409	551	225359	7	red	0.8983395
6	368 x 496	368	496	182528	6	toy	0.896281
7	331 x 446	331	446	147626	6	toy	0.77030724
8	298 x 401	298	401	119498	5	toy	0.7545809
9	268 x 361	268	361	96748	4	toy	0.90437394
10	241 x 325	241	325	78325	3	toy	0.8983962
11	217 x 293	217	293	63581	3	toy	0.7133052
12	195 x 264	195	264	51480	3	toy	0.73693734
13	176 x 238	176	238	41888	2	toy	0.70034945
14	158 x 214	158	214	33812	3	toy	0.71050787
15	142 x 193	142	193	27406	1	toy	0.6792168
16	128 x 174	128	174	22272	2	toy	0.63405913
17	115 x 157	115	157	18055	2	toy	0.634116
19	94 x 127	94	127	11938	1	machine	0.72266835

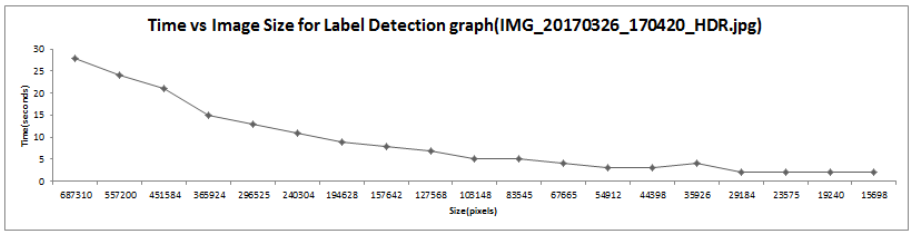
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *toy car* word will display in their mind first. But the label detection detected *toy* description. Therefore, the minimum best image size display is in *104px x 141px* and toy description is fall on *1* seconds with the score *0.6552992.*



IMG_20170317_171156.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	480 x 648	480	648	311040	6	skin	0.76968384
2	432 x 583	432	583	251856	5	skin	0.7723425
3	389 x 525	389	525	204225	5	skin	0.7722683
4	350 x 473	350	473	165550	3	skin	0.7675255
5	315 x 426	315	426	134190	6	product	0.7629654
6	284 x 383	284	383	108772	4	product	0.7651249
7	256 x 345	256	345	88320	2	product	0.7610888
8	230 x 311	230	311	71530	3	skin	0.7710263
9	207 x 280	207	280	57960	1	skin	0.7754823
10	186 x 252	186	252	46872	2	product	0.7696241
11	167 x 227	167	227	37909	2	skin	0.7800545
12	150 x 204	150	204	30600	2	skin	0.75959855
13	135 x 184	135	184	24840	1	product	0.7614943
14	122 x 166	122	166	20252	2	candle	0.7711818
15	110 x 149	110	149	16390	1	product	0.7495214

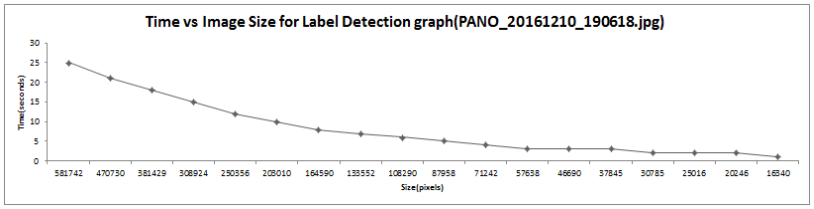
The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *bottle glass* word will display in their mind first. But the label detection detected *product* description. Therefore, the minimum best image size display is in *99px x 134px* and product description is fall on *1* seconds with the score *0.758632.*



IMG_20170326_170420_HDR.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	622 x 1105	622	1105	687310	28	human action	0.96107084
2	560 x 995	560	995	557200	24	human action	0.96047467
3	504 x 896	504	896	451584	21	human action	0.9593854
4	454 x 806	454	806	365924	15	human action	0.9600551
5	409 x 725	409	725	296525	13	human action	0.96120256
6	368 x 653	368	653	240304	11	human action	0.95908964
7	331 x 588	331	588	194628	9	human action	0.9606153
8	298 x 529	298	529	157642	8	human action	0.9608691
9	268 x 476	268	476	127568	7	human action	0.9606138
10	241 x 428	241	428	103148	5	human action	0.95933485
11	217 x 385	217	385	83545	5	human action	0.9582973
12	195 x 347	195	347	67665	4	human action	0.9615941
13	176 x 312	176	312	54912	3	human action	0.9583797
14	158 x 281	158	281	44398	3	human action	0.954784
15	142 x 253	142	253	35926	4	human action	0.9553834
16	128 x 228	128	228	29184	2	human action	0.95603096
17	115 x 205	115	205	23575	2	dish	0.90950185
18	104 x 185	104	185	19240	2	food	0.8992839

The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *eating pizza* word will display in their mind first. But the label detection detected *dish* description. Therefore, the minimum best image size display is in *94px x 167px* and dish description is fall on *2* seconds with the score *0.8921192.*



PANO_20161210_190618.jpg

#	Image Size	Width	Height	Size(pixels)	Time(Second	Description	Score
1	1026 x 567	1026	567	581742	25	sky	0.9570402
2	923 x 510	923	510	470730	21	town	0.8257609
3	831 x 459	831	459	381429	18	sky	0.83253574
4	748 x 413	748	413	308924	15	sky	0.8296607
5	673 x 372	673	372	250356	12	structure	0.86242706
6	606 x 335	606	335	203010	10	structure	0.8434053
7	545 x 302	545	302	164590	8	structure	0.8392918
8	491 x 272	491	272	133552	7	structure	0.8409311
9	442 x 245	442	245	108290	6	structure	0.8573149
10	398 x 221	398	221	87958	5	structure	0.90394276
11	358 x 199	358	199	71242	4	structure	0.84792995
12	322 x 179	322	179	57638	3	structure	0.87340903
13	290 x 161	290	161	46690	3	structure	0.879545
14	261 x 145	261	145	37845	3	photography	0.8439305
15	235 x 131	235	131	30785	2	photography	0.85388446
16	212 x 118	212	118	25016	2	skyline	0.9115596
17	191 x 106	191	106	20246	2	skyline	0.8917798

The result of this testing is to show the best minimum image size that is processing label detection within photo with the minimum time processing by reducing 10% each time. Usually when user sees the photo, the *skyline* word will display in their mind first. But the label detection detected *skyline* description. Therefore, the minimum best image size display is in *172px x 95px* and skyline description is fall on *1* seconds with the score *0.8775325*.

Eidetic Search

ORIGINALITY REPORT

%11	%9	%4	%10
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universiti Tunku Abdul Rahman Student Paper	%9
2	eprints.utar.edu.my Internet Source	%1
3	Submitted to Informatics Education Limited Student Paper	<%1
4	www.worldrailfans.org Internet Source	<%1
5	www.cnet.com Internet Source	<%1
6	eprints.dinus.ac.id Internet Source	<%1

EXCLUDE QUOTES	OFF	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	OFF		