GOOGLE CLOUD VISION FOR UNITY

Version 1.03

Created by PaulArt

Visit PaulArt.cc for more info

If you have any questions contact me via email

paulartcc@gmail.com

Contents

Overview

Google Cloud Vision for Unity is an asset that allows you to use the full power of Google Cloud Vision intelligence API.

It includes 11 different object recognition types:

- Detect text in images
- Detect handwriting in images
- Detect crop hints
- Detect faces
- Detect image properties
- Detect labels
- Detect landmarks
- Detect logos
- Detect multiple objects
- Detect explicit content (safe search)
- Detect Web entities and pages

Google Cloud Vision documentation
https://cloud.google.com/vision/docs/

Unity engine documentation
http://docs.unity3d.com/Manual/index.html

Required Packages

ONLY FOR 1.02 VERSION AND BELOW

For proper work this asset requires additional packages. You can download then with Asset Store for free.

JsonFormatter by Bayat Games

https://assetstore.unity.com/packages/tools/input-management/jsonformatter-97094

Runtime File Browser by Süleyman Yasir Kula

https://assetstore.unity.com/packages/tools/gui/runtime-file-browser-113006

If you have any troubles with assets obtaining or installation please contact me for support.

How to Implement

The project can be open in various versions of Unity, but I suggest you to use any 2019 edition. It can be downloaded here:

https://unity3d.com/get-unity/download/archive

After Unity is installed, import the project into your workplace and open the Main scene.

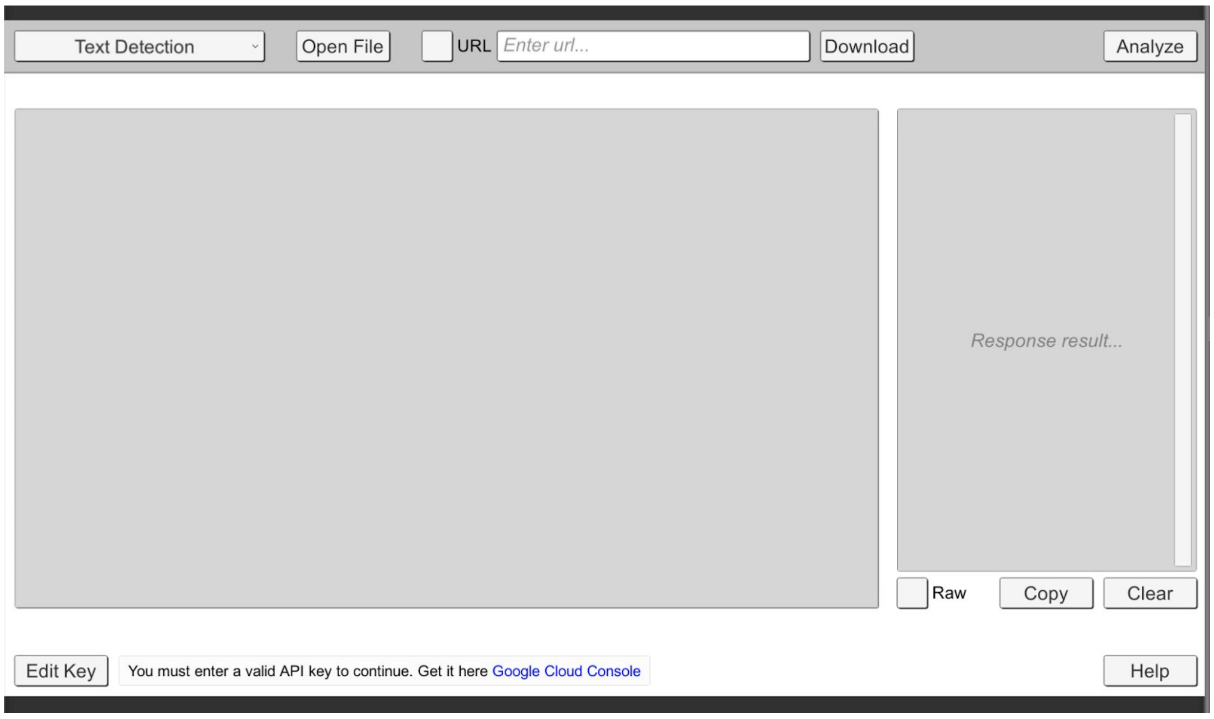
You have to get an API key first.

You can get it here
https://cloud.google.com/compute/docs/console

This documentation does not include a step by step guide about how to receive Google Cloud Vision API key.

How to Use

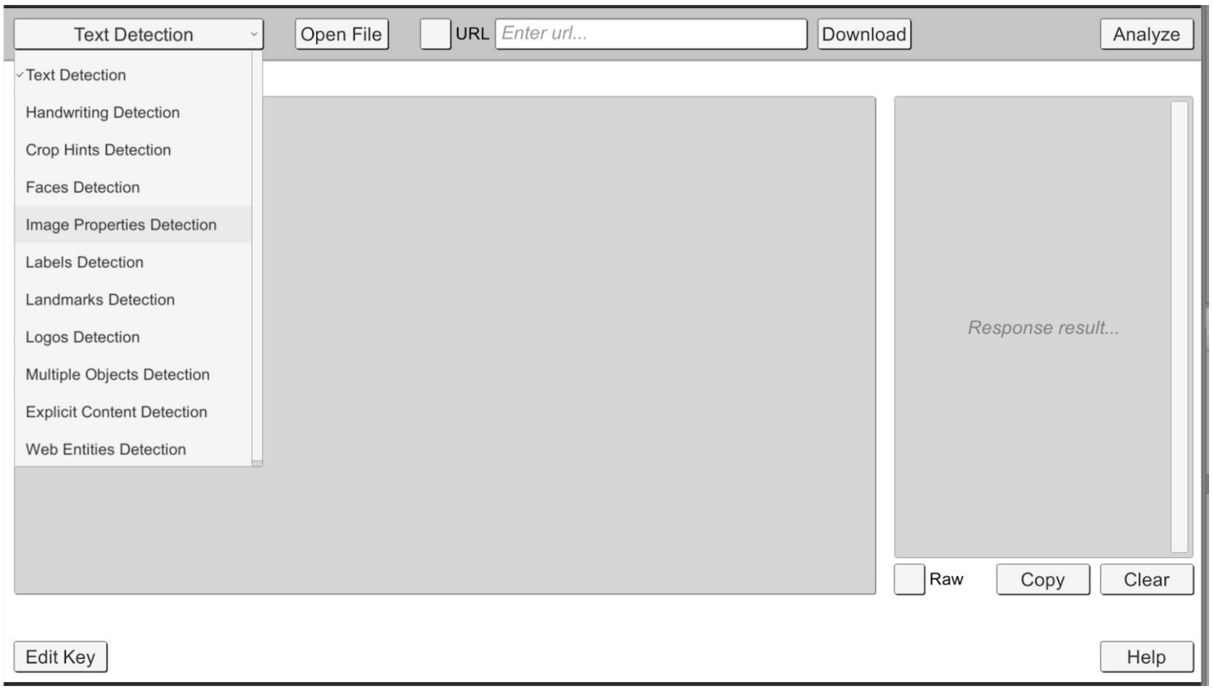
Once you've launched the program you will this window



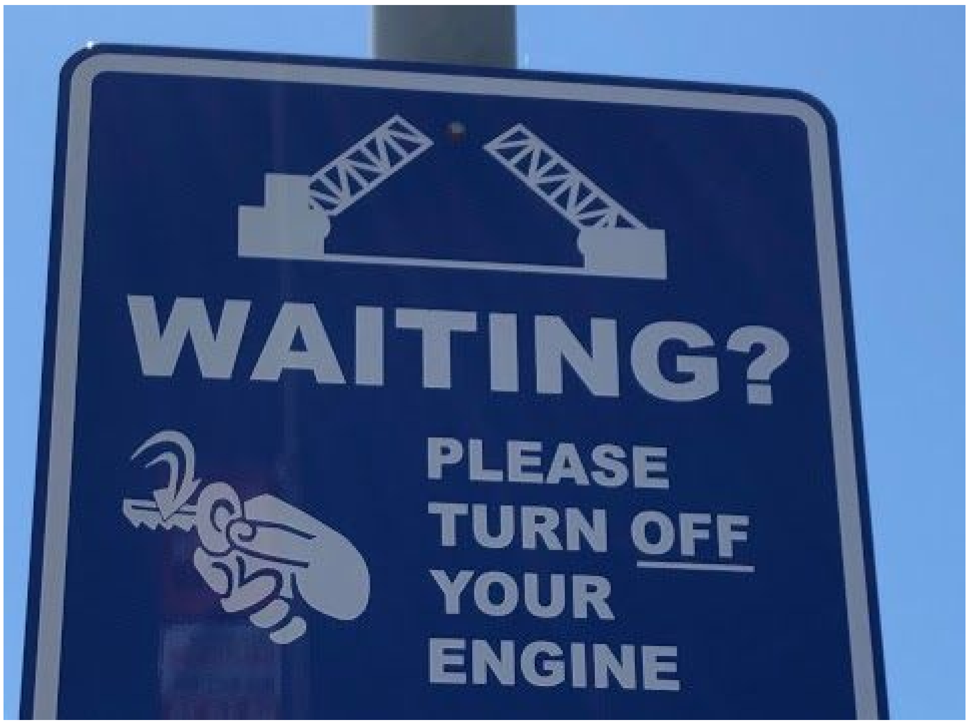
You must enter a valid API key to continue. Get it here Google Cloud Console

First of all you have to enter a valid API key to continue. Click the "Edit Key" button and type it into the field. Then press the "Save" button.

Then select the required detection type at the bottom left dropdown.

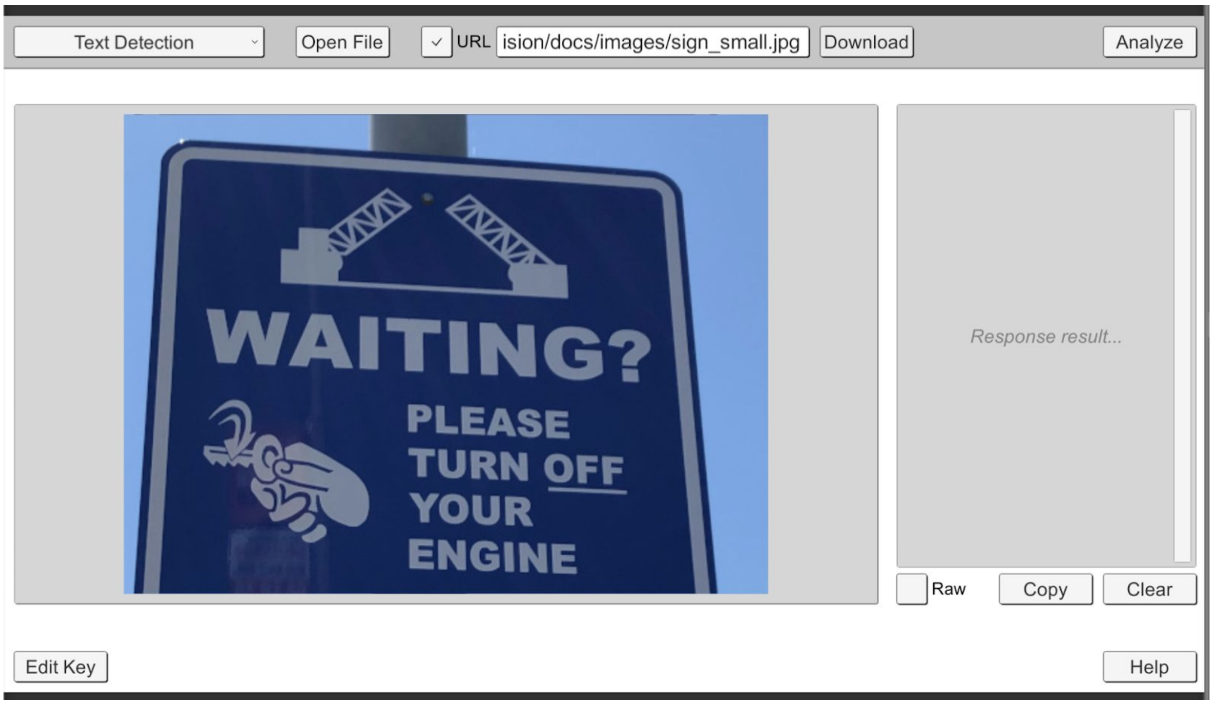


For this example we are going to use basic Text Detection with this sample image.



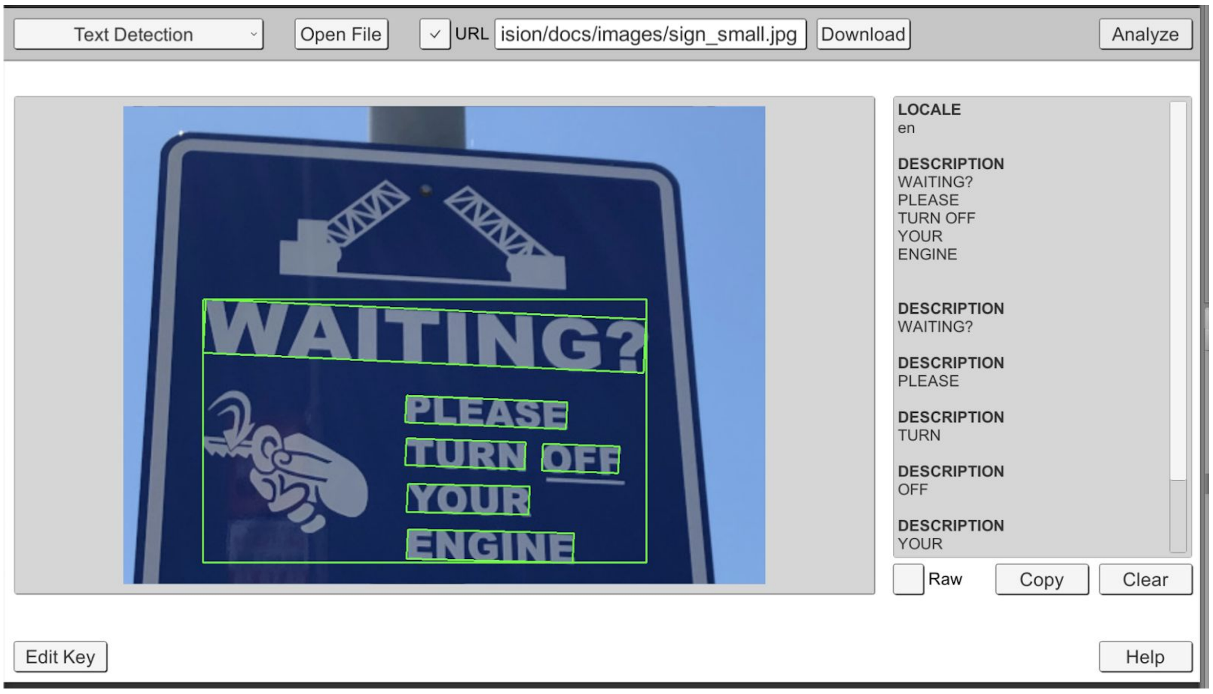
At the next step you have two options. You can download the image from here https://cloud.google.com/vision/docs/images/sign_small.jpg and open it via file brower clicking the "Open File" button or you can use the URL path instead.

Let's follow the second way. Click at the "URL" checkbox to say the program we are going to use web images. Then enter url link and press "Download" button.



If your Internet connection is on and at the time you are reading this guide link is still valid you will see a downloaded image on your screen.

Then just press “Analyze” button and wait until it’s done.



LOCALE
en

DESCRIPTION
WAITING?
PLEASE
TURN OFF
YOUR
ENGINE

DESCRIPTION
WAITING?

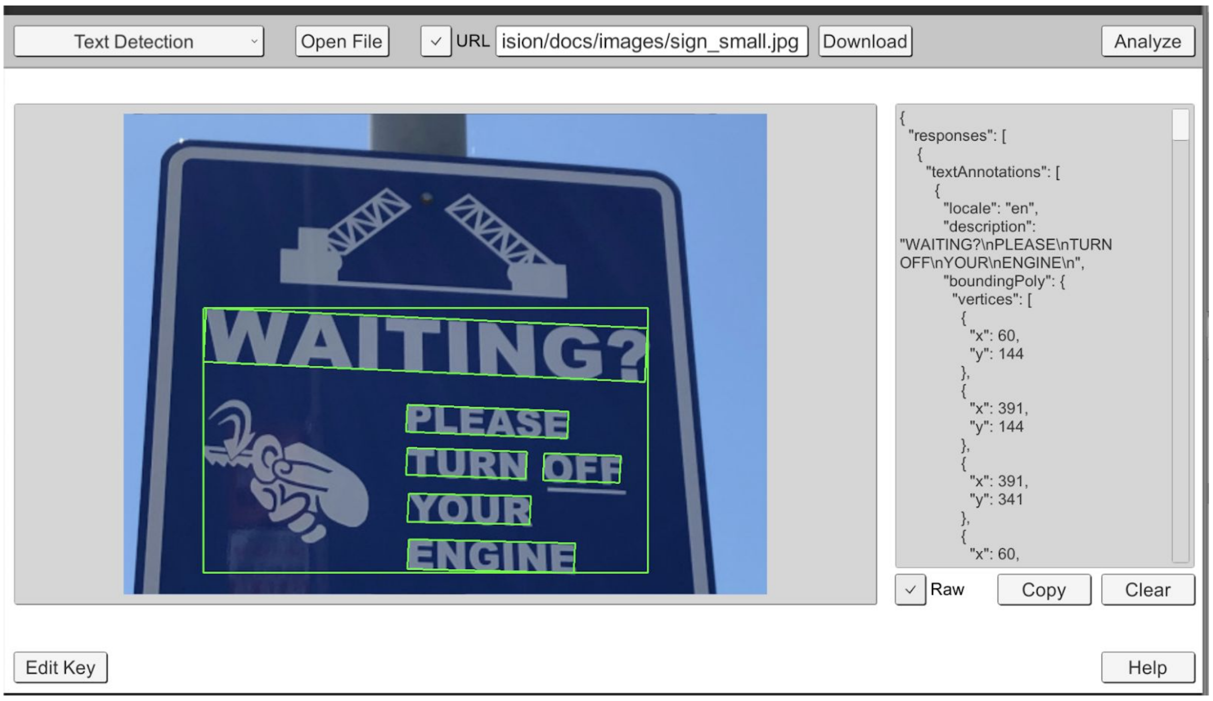
DESCRIPTION
PLEASE

DESCRIPTION
TURN

DESCRIPTION
OFF

DESCRIPTION
YOUR

If you want to see a raw unformatted response then mark “Raw” checkbox.



```
{
 "responses": [
  {
   "textAnnotations": [
    {
     "locale": "en",
     "description":
"WAITING?\nPLEASE\nTURN
OFF\nYOUR\nENGINE\n",
     "boundingPoly": {
      "vertices": [
       {
        "x": 60,
        "y": 144
       },
       {
        "x": 391,
        "y": 144
       },
       {
        "x": 391,
        "y": 341
       },
       {
        "x": 60,
```

That’s it!

Functions Listing

Main.cs

void Start()

Populate dropdown with detection options. This values are based on TAGs variables from each detection model.

void Update()

Update loading spinner and result text placeholder.

public void BrowseFiles()

Open file browse window. Returns path if image is selected and sends it further.

private void RemoveRectangles()

Delete rectangles from the screen.

public void Analyze()

This function starts the selected detection type process. It uses detection TAG and value from dropdown to determine which one is selected.

private void NetworkError(string error)

Simple function to print different errors

private bool CheckImageSource()

Check if image is loaded and ready to be analyzed.

private void ShowResult(List<Vertices[]> vertices, List<string[]> values, string rawData)

Bridge function. Receives parsed server response and sends it further.

private void DisplayText(List<string[]> values)

Display parsed respone data with text component.

public void OpenImage(string path)

Load image from storage using path.

public void DownloadImage()

Start the async process to download image from the web.

public void ShowImage(Texture2D texture, bool isDownload)
Bridge function. Receives opened image and sends it further.

private void ShowLoading(bool show)
Turn on/off loading spinner.

public void CopyResult()
Copy result text to clipboard.

public void ClearResult()
Clear text from result container.

public void ShowRawResult(Toggle toggle)
Display text result as original json response.

ApiKey.cs

void Start()
Load API key from storage

public void OpenEditKey()
Reveal or close API key input field.

public void SaveKey()
Save API key to storage.

private void CheckKey()
Check if API key is set.

public void OpenCloudConsole()
Open Google Cloud webpage in browser.

public void ShowHelp()
Show text hints about the program.

Utilities.cs

public static Texture2D OpenImage(string path)

Load image from storage using path.

public static string ConvertToBase64(string path)

Convert image to base64 representation.

public static IEnumerator DownloadImage(Action<Texture2D, bool> onSuccess, Action<string> onFailure, string image_url)

Download image from the Internet using the provided URL.

public static string GenerateRequest(string request_path, string image_path, bool isUrl)

Get the request body from the file and fill it with required parameters.

public static bool CheckInternetConnection()

Check if Internet connection is available.

public static bool IsApiKeySet()

Check if the API key is set.

Storage.cs

public static string LoadKey()

Load saved API key from PlayerPrefs

public static void SaveKey(string key)

Save API key to PlayerPrefs.

ImageProcessing.cs

public static void DrawRectangles(List<Vertices[]> vertices, UILineRenderer rectPrefab, GameObject container, Image image)

This is the most important function. It takes an array of vertices (x, y coordinates) and draws rectangles side by side.
For every detected object we receive four coordinates.
To draw lines we have to do some math transformations according to resized image width and height and container pivot.

public static void ShowImage(Texture2D texture, Image image, GameObject container)

This function is used to adjust selected images to container size. It resizes the image so it fits any screen area.
It's important to save original image width and height somewhere as also new values.

RequestHandler.cs

public static IEnumerator SendRequest(string requestString)

Send web request with provided params to the selected URL.

public static bool CheckResult()

Check if the response exists.

public static string GetResult()

Get response data.

public static string GetError()

Get response error text.

DetectionHandler.cs

public static IEnumerator Run(Action<List<Vertices[]>, List<string[]>, string> onSuccess, Action<string> onFailure, string image_path, bool isUrl)

Generate the request body according to the selected detection type model and send it to Google Cloud server.
Receive response and parse it into coordinates and text information.