How to User Marker less AR Windows demo

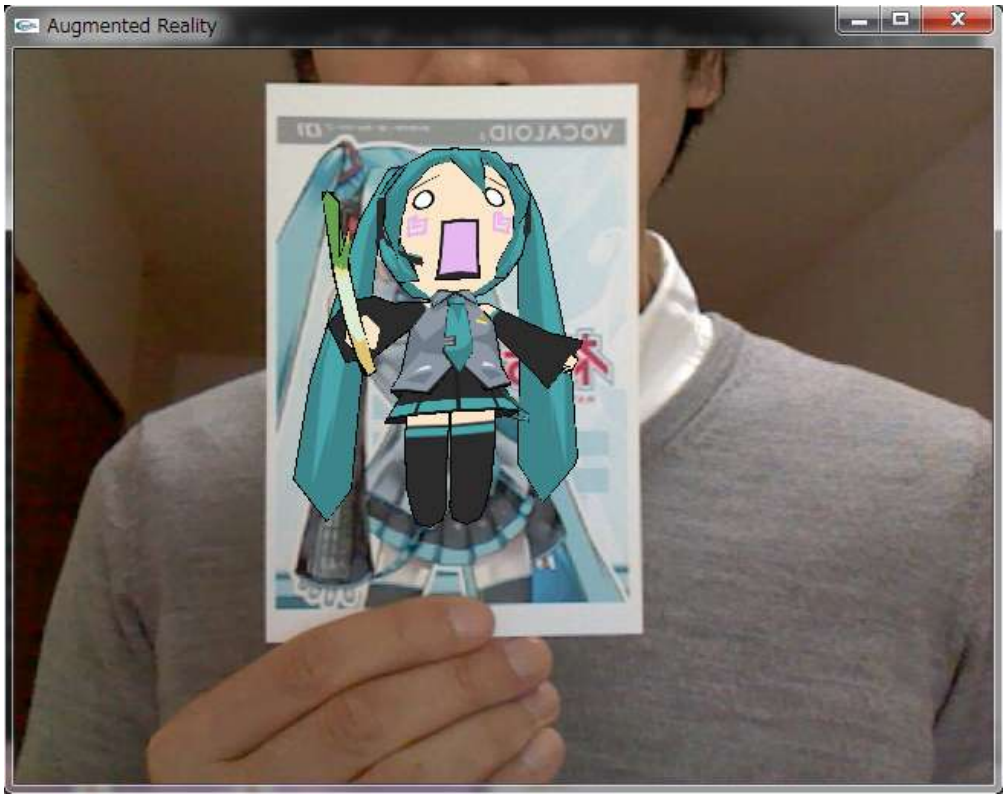
2012/01/07

Takuya MINAGAWA (z.takmin@gmail.com)

1 Overview

This program is so called “Marker less Augmented Reality” application that recognizes an arbitrary image captured by USB camera and overlays 3D model over it.

This document describes how to use marker less AR Windows demo application “ARengine.exe”. This program is written in C++ using OpenCV 2.3.1 and GLUT 3.7.6 so I believe you can build this program on the other platform, like Linux, Mac, etc.



2 Set Up

VC++2010 Runtime

To run this program, Visual C++ 2010 redistributable package must be installed. If you have not installed it, you can get it at the web site below:

For x86:
http://www.microsoft.com/downloads/details.aspx?FamilyID=a7b7a05e-6de6-4d3a-a423-37bf0912db84&displayLang=ja

For x64:
http://www.microsoft.com/downloads/details.aspx?FamilyID=bd512d9e-43c8-4655-81bf-9350143d5867&displayLang=ja

For IA64:
http://www.microsoft.com/downloads/details.aspx?FamilyID=1a2df53a-d8f4-4bfe-be35-152c5d3d0f82&displayLang=ja

Camera Calibration

If you want to overlay 3D model over an image precisely, you need to calculate internal parameters of camera previously. The way of camera calibration is described later in this document. A calibration file for Logicool QCAM Pro9000 (Image resolution is 640x480) is in this demo and you can use this file if you use a similar spec camera.

Printing Marker Image

Images to recognize in this demo application are in "marker" folder. Print them before you use this program.

Start AR program

Double click "ARstart.bat", then the application starts and a window of camera image opens. Show a printed image to the camera, then you can see 3D model overlay in a window.

Key functions are as below:

ESC or Q : Stop the application

F : Exchange full screen mode

3 Configuration File

You can change some functionalities of this demo by editing “config.xml”.

Here is a sample:

```
<?xml version="1.0"?>
<opencv_storage>
<VisualWord>
   <visualWord>ardemo/visualWord.bin</visualWord>
   <index>ardemo/vw_index.bin</index>
</VisualWord>
<ObjectDB>ardemo/db.txt</ObjectDB>
<camera_matrix>ardemo/camera_matrix_qcam.txt</camera_matrix>
<focal_length>0.5</focal_length>
<max_query_size>320</max_query_size>
<full_screen_mode>false</full_screen_mode>
<mirror_mode>true</mirror_mode>
<model_info>
   <_>
      <id>1</id>
      <ModelType>MQOSEQ</ModelType>
      <modelfile>ardemo/miku/miku.xml</modelfile>
      <scale>0.002</scale>
            <init_pose>
               <!-- z-y-x euler angle -->
               <yaw>0</yaw>
               <pitch>0</pitch>
               <roll>0</roll>
               <x>0</x>
               <y>-0.2</y>
               <z>0</z>
            </init_pose>
   </_>
   <_>
      <id>2</id>
      <ModelType>METASEQ</ModelType>
      <modelfile>ardemo/ninja.mqo</modelfile>
```

```
    <scale>0.002</scale>
          <init_pose>
            <!-- z-y-x euler angle -->
            <yaw>0</yaw>
            <pitch>0</pitch>
            <roll>0</roll>
            <x>0</x>
            <y>-0.2</y>
            <z>0</z>
          </init_pose>
  </_>
  <_>
    <id>3</id>
    <ModelType>SLIDE</ModelType>
    <modelfile>ardemo/slides/slide1.xml</modelfile>
    <scale>0.004</scale>
  </_>
</model_info>
<WaitingModel>
  <timer>60</timer>
  <ModelType>SLIDE</ModelType>
  <modelfile>ardemo/waiting.xml </modelfile>
  <scale>0.005</scale>
          <init_pose>
            <!-- z-y-x euler angle -->
            <yaw>0</yaw>
            <pitch>0</pitch>
            <roll>0</roll>
            <x>0</x>
            <y>0</y>
            <z>0</z>
          </init_pose>
</WaitingModel>
</opencv_storage>
```

<VisualWord>
File path of a feature point dictionary and its index file.　If it is a XML/YAML format file, only a feature dictionary file path must be set to <visualWord> tag.　If it is a binary format file, both of <visualWord> and <index> tags must be set.　The way to create dictionary file is described later in this document.

<ObjectDB>
File path of a marker image database.　The way to create image database is described later in this document.

<camera_matrix>
File path of internal camera parameter.　The way to calibrate camera is described later in this document.

<focal_length>
Virtual focal length of a camera: this parameter is used for rendering 3D model over camera image.　If a part of 3D model, that is near from a camera, can't be displayed, set this parameter smaller.

<max_query_size>
An image size to recognize:　a captured image is resized to be smaller than this size before recognition.　The smaller you set this parameter, the faster recognition time. Instead, the closer you need to hold a marker image to a camera.

<full_screen_mode>
If this parameter is set "true", a camera captured image is displayed on full screen at the time of startup.

<mirror_mode>
If this parameter is set "true", a camera captured image is displayed mirror-reversed.

<model_info>
Definition of relationships between a marker image and a 3D model.

<id>
ID of registered marker image.

<ModelType>

A type of 3D model: there are 3 types of 3D model in this version.

- METASEQ: MQO (Metasequoia) 3D model file
- MQOSEQ: sequential MQO file (For Animation)
- SLIDE: Showing images by slide show

<modelfile>

A configuration file of 3D model, which depends on ModelType.

- METASEQ: Write a file path to MQO file.
- MQOSEQ: Write a file path to a configuration file of sequential MQO. In this file, the file name header and the number of MQO files are specified. This file must be placed in the same directory of sequential MQO files.
- SLIDE: Write a file path to a configuration file of slide show. In this file, image file paths and their sequence to show are specified.

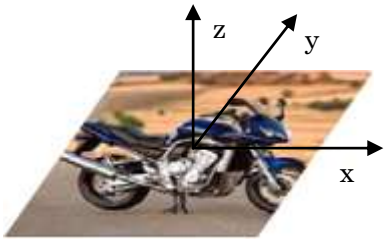
<scale>

Scale of 3D model. You can indicate the size of 3D model to display by this parameter.

<init_pose>

Initial position and pose of 3D model to display on a marker. A position is represented as translation parameter (X, Y, Z) and a pose is represented as rotation degree parameter (yaw/pitch/roll) in a marker coordinate. The definition of a marker coordinate is shown below:

Registered Image



Marker Coordinate

Rotation parameter (yaw/pitch/roll) is the same as Z-Y-Z Euler angle: 3D model is rotated around Z, Y, X respectively (degree).

<WaitingModel>

Setting of waiting 3D model. You can display 3D model on window when no marker

images are appeared for a specified time.　For instance, you can display welcome messages or instruction here.　The way to configure waiting model is almost the same as <model_info>, though multiple models cannot be set here.　At <timer> tag, you can specify a number of frames; a waiting model is displayed when no marker image is recognized for this number of frames.
If <WaitingModel> tag is not appeared in a configuration file, a waiting model is never displayed.

4 Camera Calibration

In order to overlay 3D model precisely, internal camera parameters can be calculated by the following instructions:

1 Print calibration board
   1.1 Print a checker pattern “tools/pattern.pdf”.
   1.2 Past it on a flat board.
   1.3 Measure the length of one block of checker pattern by ruler.
2 Set up a USB camera to a PC.
3 Start “ARengine.exe” by double click, then a console window is opened.
4 Obtain images for calibration from USB camera.
   4.1 At the “command:“ prompt on console window, enter “get_camera_images” and push enter key.
   4.2 At the prompt “number of images:“, enter a number of images you want to save. 10-30 images are recommended (max. 50).
   4.3 At the prompt “frame interval:“, enter an interval frame number. For instance, when you set this frame interval “30”, then camera captured image is saved in HDD at every 30 frame.
   4.4 At the prompt “save directory:“, enter folder path where you want to save images.
   4.5 At the prompt “file header:“, enter the header of file name of captured images to save, then camera captured image is displayed on a new window.
   4.6 Place the calibration board in front of camera. All checker blocks on the board must be displayed in a window.
   4.7 Push “s” button then image capture starts, and move the checker board slowly. When the specified numbers of image are saved, the captured image window is closed automatically.
5 Calibrate camera using saved images.
   5.1 At the prompt “command:”, enter “camera_calibration”.
   5.2 At the prompt “checkerboard image list:”, enter the file path of “imglist.txt” which is saved in the folder specified in 4.4.
   5.3 At the prompt “checkerboard row number:“, enter the number of checker corner in a row. For instance, enter “10” when “patter.pdf” has been captured vertically, and enter “7” otherwise.
   5.4 At the prompt “checkerboard col number:“, enter the number of checker corner in a column. For instance, enter “7” when “pattern.pdf” has been

captured vertically, and enter “10” otherwise.

5.5 At the prompt “checker size(mm):”, enter the length of 1 block of checker pattern measured at 1.3, then the calculation of the internal camera parameter starts

5.6 At the prompt ”camera parameter file name:” after calculation of camera parameter, enter the file path (directory name + file name) to save the calculated parameter file. A camera parameter is saved as XML or YAML file format. You can specify the file format by the extension of file path.

6 Enter “exit”, then finish “ARengine.exe”.

7 Set the file path camera parameter to <camera_matrix> in “config.xml”.

5 How to register marker images

This engine includes a function of object recognition that recognizes a marker image and returns its id number. Using this functionality, a 3D model to be displayed can be changed with respect to each marker image.
This chapter describes how to register marker images to object recognition function.

1 Prepare images to recognize (image group A). If the number of images is small, prepare images not to recognize (image group B). There are no criteria for this number, but total number of images (A+B) is better to be more than 15 empirically. If total number of images is too small, its recognition accuracy is not good. If you want to reduce false positive, increase the number of B.

2 Create a text file (list file C), in which image file paths of both groups are listed.

ex.
img/A1.jpg
img/A2.jpg
img/A3.jpg
img/B1.jpg
img/B2.jpg
img/B3.jpg
.
.
.

3 Create a text file (list file D), in which image file paths of group A are listed. You can define ID to each image by inserting its ID number and comma into the top of each line. When you don't define ID in this list file, its ID number is assigned automatically. This ID is the marker ID, which must be written in <id> tag in <model_info> tag in "config.xml".

ex.
1,img/A1.jpg
2,img/A2.jpg
3,img/A3.jpg

4 Start "ARengine.exe" by double click.

5 At the prompt "command:", enter "create_vw" then you can create feature dictionary file and its index file.

5.1 At the prompt "img file list:", enter file path of C, then creation of a feature

dictionary and index starts.

6 At the prompt “command:”, enter "save_vw" then you can save a feature dictionary file and its index file. A feature dictionary and its index are saved as XML or YAML format. Enter “save_vw_bin”, if you want to save a feature dictionary as binary format that is separated from its text format index file. Using binary format, you can load and save dictionary file more rapidly and can make file size smaller. These names of saved dictionary and its index file must be written in <VisualWord> tag in “config.xml”.

6.1 At the prompt “feature dictionary file:”, enter a file name to save. If it is not binary mode, you can choose file format (XML or YAML) by a file extension. For example, if its file extension is “.xml”, then this file is saved as XML, and otherwise it is saved as YAML. If its file extension is like “.txt.gz” or “.xml.gz”, then it is saved with compression.

6.2 At the prompt “index file:”, enter a index file name to save. (Binary mode only)

7 At the prompt “command:”, enter "registf" then you can register images to recognize.

7.1 At the prompt “regist list file:”, enter the file name of D, then registered image id is printed on a standard output.

8 At the prompt “command:”, enter “save_object_DB” then you can save registered image database. This database file must be written in <ObjectDB> tag in “config.xml”.

8.1 At the prompt “save file name:”, enter file name to save.

9 At the prompt “command:”, enter “exit” then stop “ARengine.exe”.

6. How to evaluate object recognition function

This chapter explains how to evaluate object recognition function alone. Object recognition function can be evaluated without USB camera, 3D model, and configuration file, but with a feature dictionary file, a database file, and query image files.

There are two methods for evaluation. One is to display the recognition result of each query image one by one, another is to output recognition results of query images into text file by batch processing.

How to load a feature dictionary and a database file

If you did not finish “ARengine.exe” after you created a feature dictionary and a database, the both have already been loaded and you can skip the following steps.

1 Start “ARengine.exe”.

2 At the prompt “command:”, enter "load_vw" to load a feature dictionary and its index. If you want to load a binary format file, enter “load_vw_bin”.

  2.1 At the prompt “feature dictionary file:“, enter a file path of a feature dictionary file you named at “6.” of the previous section.

  2.2 At the prompt “index file:”, enter an index file name of the feature dictionary.

3 At the prompt “command:”, enter "load_objectDB" to load a database file.

  3.1 At the prompt “load file name:”, enter a database file path you named at “7.” Of the previous section.

Single Image Recognition Test

Before single image recognition test, you need to load a feature dictionary and a database.

1 Prepare a query image file (bmp, jpg, png, etc).

2 At the prompt “command:”, enter “query” to do single image recognition.

  2.1 At the prompt “query image file:”, enter file path of a query image.

3 If a query image matches to the one in the database, a new window is opened and the position of object is displayed. The matched object ID is also displayed in the console window.

Batch Image Recognition Test

Before batch image recognition test, you need to load a feature dictionary and a a database.

1 Create a text file (list file E), in which query image file paths are listed.

2 At the prompt “command:”, enter "queryf" to do batch image recognition test.

2.1 At the prompt "img file list:", enter the file path of E.

2.2 At the prompt "result file:", enter a file path, into which the recognition results are written.

3 The recognition results are output into file E as a CSV format. If a query image matches in a registered object, then its ID is written, otherwise “-1”.

7. A List of commands of ARengine.exe

create_vw	Create a feature dictionary and its index.
save_vw	Save a feature dictionary and its index into a file.
load_vw	Load a feature dictionary and its index from a file.
save_vw_bin	Save a feature dictionary into a binary format file and its index into a XML/YAML file.
load_vw_bin	Load a feature dictionary from a binary format file and its index from a XML/YAML format file.
registf	Registered multiple images into a database by batch process.
regist	Registered one image into a database.
remove	Remove one image from a database by indicating the ID.
clear	Clear all images from database.
save_objectDB	Save the current registered image information into a file.
load_objectDB	Load a database from a file.
query	Recognize one image from a file.
queryf	Recognize multiple images and output their results into a file.
get_camera_images	Capture and save sequential images from camera.
camera_calibration	Calculate internal camera parameters.
ar_start	Start AR function.