ECE 2nd

PICTURE MATCHING WITH LABVIEW

SUBMMITTED BY:

PIYUSHA SAYAL (18105074) AND MUSKAN MITTAL (18105062)

CONTENT

What Is LabVIEW?

LabVIEW (**Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench) is a graphical programming environment which has become prevalent throughout research labs, academia and industry. It is a powerful and versatile analysis and instrumentation software system for measurement and automation. Its graphical programming language called *G programming* is performed using a graphical block diagram that compiles into machine code and eliminates a lot of the syntactical details. LabVIEW offers more flexibility than standard laboratory instruments because it is software based.

Different labview Modules are available,the one we used for this project is Vision Development module.

The Vision Development Module helps you develop software for machine vision and image processing applications. You can use it with the LabVIEW and LabVIEW NXG graphical programming environment, C, C++, and C# for Windows systems and LabVIEW for real-time systems. Choose from hundreds of image processing algorithms including filters, morphologies, pattern matching, and classification.

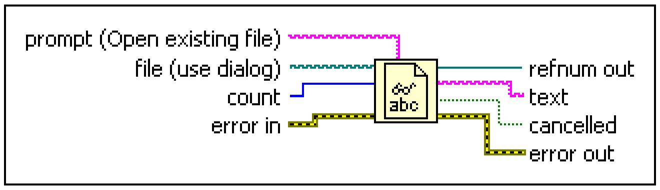
Use the Vision Development Module to take an open and adaptable approach to machine vision software development. You can choose the right hardware for your application and configure cameras, acquire images, and analyze inspection results to build fully customized machine vision systems.

Overview

Picture Matching is used to match two input images entered by the user. In order to create a more user friendly environment and making this game more interactive, some text based files are converted to speech using Text to speech Synthesis. Also speech recording and generating waveforms for the same. Images are converted to pixels of arrays which are further compared to check if they are equal or not. Vision development Module is used to read image file and typecasting images into correct format suitable for other IMAQ functions is also done. Image to array conversion gives us the array of pixels which help us in comparing the two images. Output is given in the form of writing a text file and also speech based interaction with the user.

Read a Text File

FUNCTIONS USED



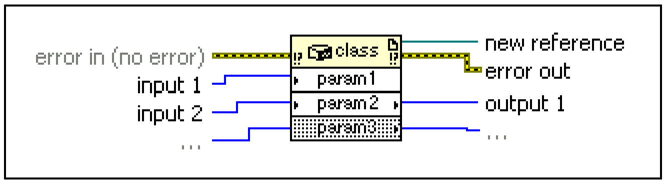
Reads a specified number of characters or lines from a byte stream file.

This function is used to read text from a file named “final.txt”. File is browsed in File path control used here.

FUNCTIONS USED

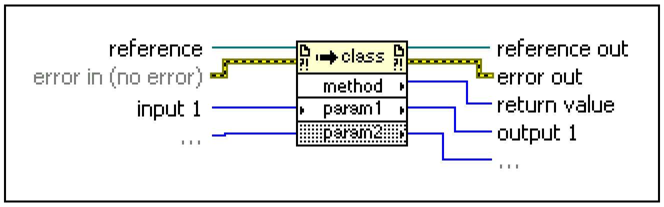
Constructor Node

Creates an instance of a .NET object. This node identifies the constructor from which to create a .NET object.



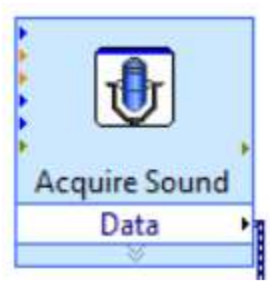
Invoke Node

Invokes a method or action on a reference. Most methods have associated parameters. If the node is configured for VI Server Application class or Virtual Instrument class and reference is unwired, reference defaults to the current Application or VI. LabVIEW includes Invoke Nodes preconfigured to access XML methods, .NET methods, and ActiveX methods.

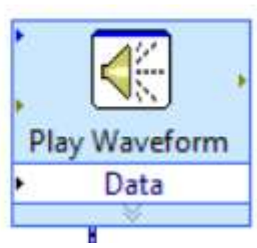


Acquire Sound

FUNCTIONS USED



Acquires data from a sound device. This Express VI automatically configures an input task, acquires the data, and clears the task after the acquisition completes.

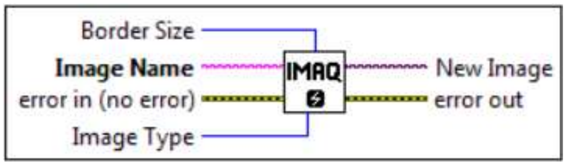


Plays data from the sound output device using finite sampling. This Express VI automatically configures an output task and clears the task after the output completes.

Acquire Sound express VI basically records sound for the time duration specified. We can play waveform with the data it extracts after recording for a given amount of time and also generate waveform for the data using Waveform Graph.

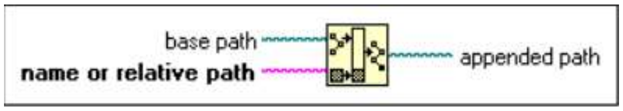
Read Image File

FUNCTIONS USED



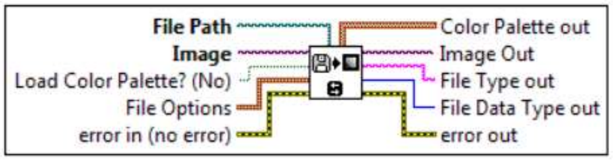
Creates a temporary memory location for an image.

- **Border Size** determines the width, in pixels, of the border to create around an image. Default value for Border size is 3.
- **Image Name** is the name associated with the created image. Each image created must have a unique name.
- **error in (no error)** describes the error status before this VI or function runs.
- **Image Type** specifies the image type.
- **New Image** is the **Image** reference that is supplied as input to all subsequent (downstream) functions used by NI Vision.
- **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information.



Creates a new path by appending a name or a relative path to an existing path.

- **base path** specifies the path to which this function appends **name or relative path**. The default is an empty path.
- **name or relative path** is the new path component appended to **base path**.
- **appended path** is the resulting path



Reads an image file. The file format can be a standard format or a nonstandard format known to the user. In all cases, the read pixels are converted automatically into the image type passed by **Image**.

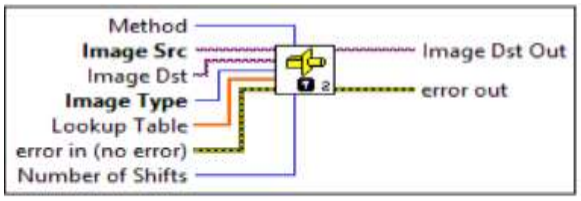
- **File Path** is the complete pathname, including drive, directory, and filename, of the file to read.
- **Image** is a reference to the image to which data from the image file is applied.

- **Image Out** is a reference to the destination image. If **Image Dst** is connected, **Image Dst Out** is the same as **Image Dst**.

IMAQ functions creates a new image by assigning a temporary location to it. An image file is entered by the user by specifying file path. In order to have complete path to the image we need to use Build path function which appends the file name to the path specified. We use IMAQ Read file function to read the image from the given path, it display us the image in the front panel with the image indicator used. We get the final image in 16 bit png format.

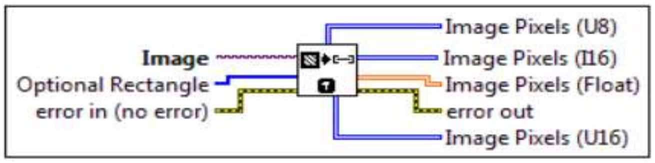
Image to Pixels Array

FUNCTIONS USED



Converts the current image type to the image type specified by **Image Type**. For 16-bit to 8-bit conversions, **Method** specifies the casting method to be used.

- **Image Src** is a reference to the source image
- **Image Dst** is a reference to the destination image
- **Image Type** specifies the image type into which the input image is converted.



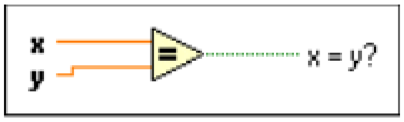
Extracts (copies) the pixels from an image, or part of an image, into a LabVIEW 2D array. This array is encoded in 8 bits, 16 bits, or floating point, as determined by the type of input image.

- **Image** is a reference to the source image.
- **Optional Rectangle** defines a four-element array that contains the left, top, right, and bottom coordinates of the region to extract. The right and bottom values are exclusive and lie outside the region. The operation applies to the entire image if the input is empty or not connected.
- **Image Pixels (U8)** returns the extracted pixel values into a 2D array. The first index corresponds to the vertical axis and the second index corresponds to the horizontal axis. The VI resizes **Image Pixels** to be the same size as **Image** or the size of **Optional Rectangle**. Use this output only when **Image** is an unsigned 8-bit image.

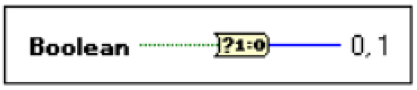
We used IMAQ Cast Image in order to convert 16 bit png file to an 8 bit image png file. This Casted image is further given as an input to IMAQ Image to Array VI to convert 8 bit image to 2D array of pixels. As we know an image is composed of matrix of pixels. To obtain matrix of pixels for the given image,we use function to do so. We can also Threshold image further into binary 2D array of pixels based on the threshold value specified.

Matching Arrays

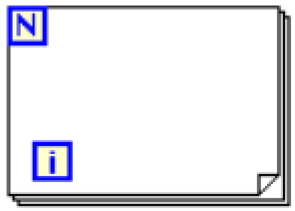
FUNCTIONS USED



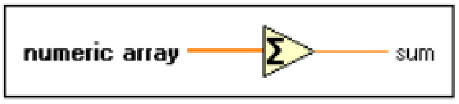
Returns TRUE if **x** is equal to **y**. Otherwise, this function returns FALSE.



Converts a Boolean FALSE or TRUE value to a 16-bit integer with a value of 0 or 1, respectively.



Executes its subdiagram n times, where *n* is the value wired to the count (**N**) terminal. The iteration (**i**) terminal provides the current loop iteration count, which ranges from 0 to *n*-1.

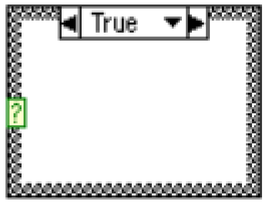


Returns the sum of all the elements in **numeric array**.

Array of pixels is matched for the images. The output returned for the equal to function is a 2D array matching values for each index of array and gives 2D array of Boolean as result. If all the values of array True we get 1 True as our output for next equal to function which consist of 2D array constant of True input only. Boolean value is converted to 1 and 0 and using for loop we get the final value for all indexes which is matched with 1 to give is the final result as a Boolean output to check whether images are matched or not.

Use Of Case Structures

FUNCTIONS USED



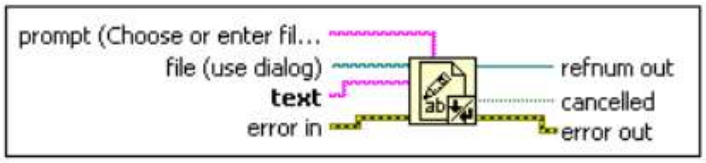
Contains one or more sub diagrams, or cases, exactly one of which executes when the structure executes. The value wired to the case selector determines which case to execute.

- **Selector label**—Displays the value(s) for which the associated case executes. You can specify a single value or a range of values
- **Subdiagram (case)**—Contains the code that executes when the value wired to the case selector matches the value that appears in the selector label
- **Case selector**—Selects which case to execute based on the value of the input data.

Boolean output of previous functions is passed as case selector input the case structure. If the output is true, "Picture Matched" is given as output. Otherwise "Picture not Matched" is given as output.

Write To a Text File

FUNCTIONS USED

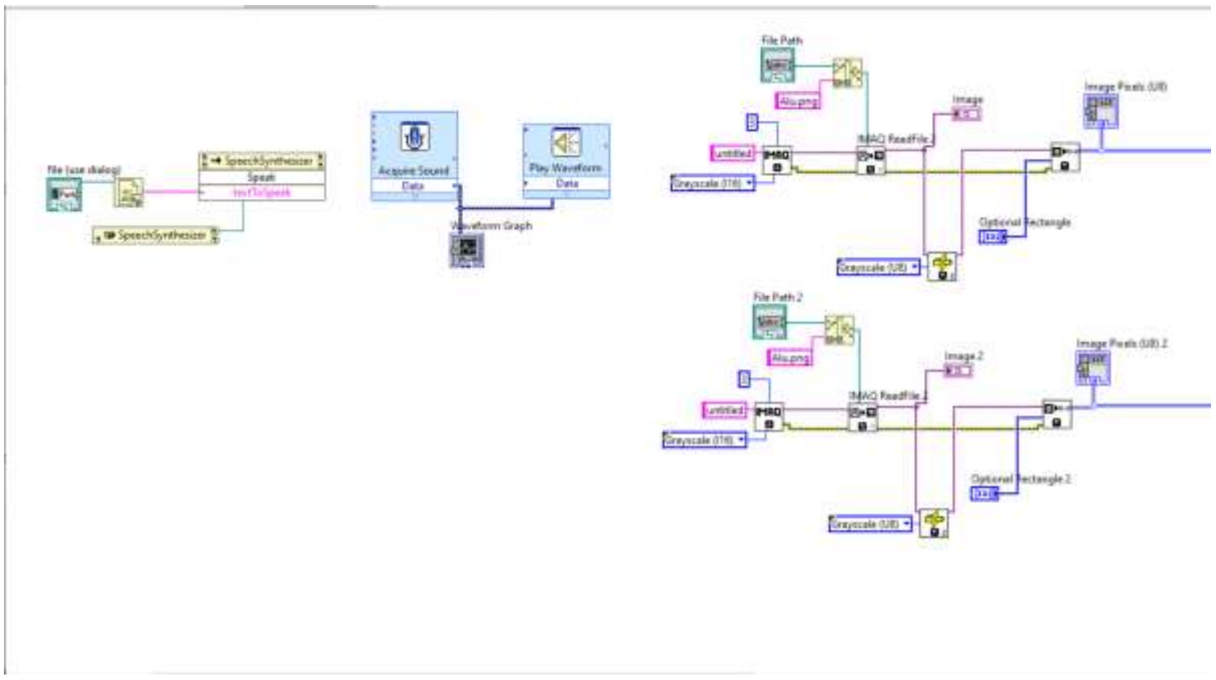


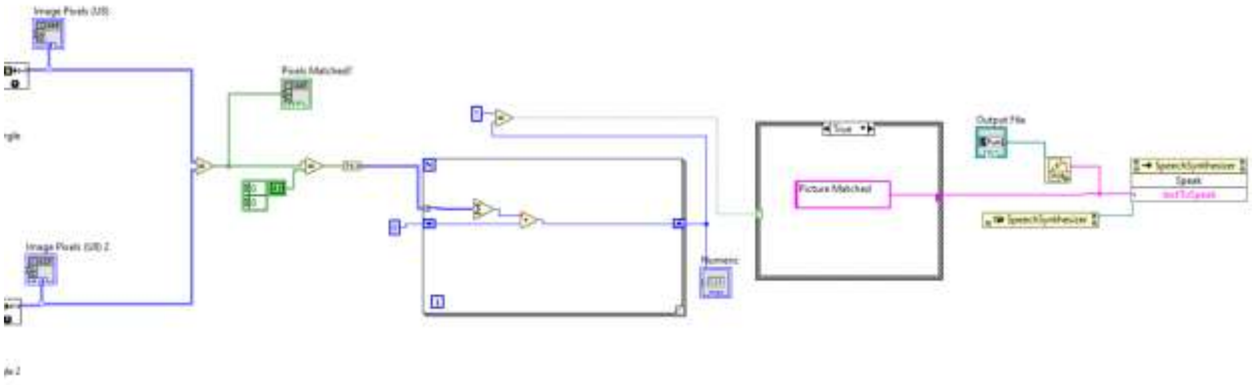
Writes a string or an array of strings as lines to a file.

The output of the case structure is written into text file. File path provided here, named as “output.txt”.

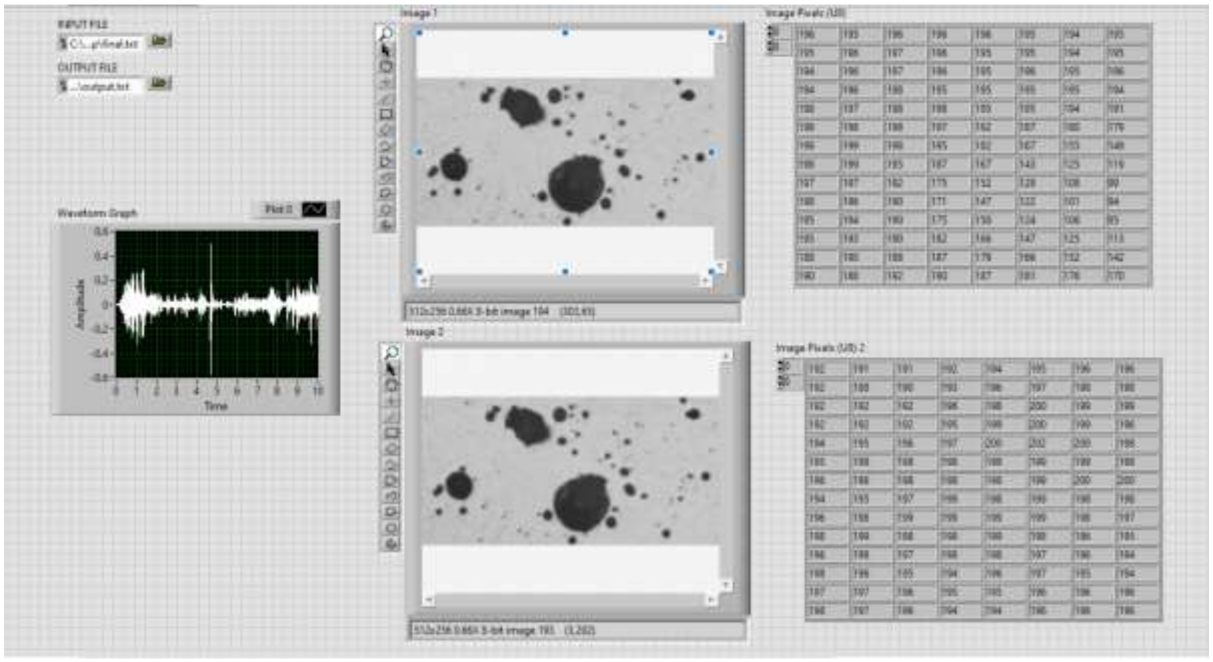
Also the text file output is converted from text to speech with the help of invoke node and constructor node. We get output in the form of sound as well as text written to the file.

BLOCK DIAGRAM





FRONT PANEL



INPUT FILE: C:\...p\final.txt

OUTPUT FILE: ...\output.txt

512x256 0.66X 8-bit image 194 (303,65)

512x256 0.66X 8-bit image 193 (3,282)

Image Pixels (U8)

196	195	196	196	196	195	194	195
195	196	197	196	195	195	194	195
194	196	197	196	195	196	195	196
194	196	198	195	195	195	195	194
198	197	198	198	195	195	194	191
198	198	198	197	192	187	180	179
198	199	198	195	182	167	155	148
198	199	195	187	167	143	125	119
197	197	192	175	152	129	108	99
198	196	190	171	147	122	101	94
195	194	190	175	150	124	106	95
195	193	190	182	166	147	125	113
188	190	188	187	179	166	152	142
190	188	192	190	187	181	176	170

Image Pixels (U8) 2

192	191	191	192	194	195	196	196
192	189	190	193	196	197	198	198
192	192	192	196	198	200	199	199
192	192	192	195	198	200	199	198
194	195	196	197	200	202	200	198
193	198	198	198	199	199	199	198
196	198	198	198	198	199	200	200
194	195	197	199	198	199	198	198
196	198	199	199	195	195	198	197
198	199	198	198	198	198	196	195
196	198	197	198	198	197	196	194
198	196	195	194	196	197	195	194
197	197	196	195	195	196	196	196
196	197	198	194	194	196	198	196

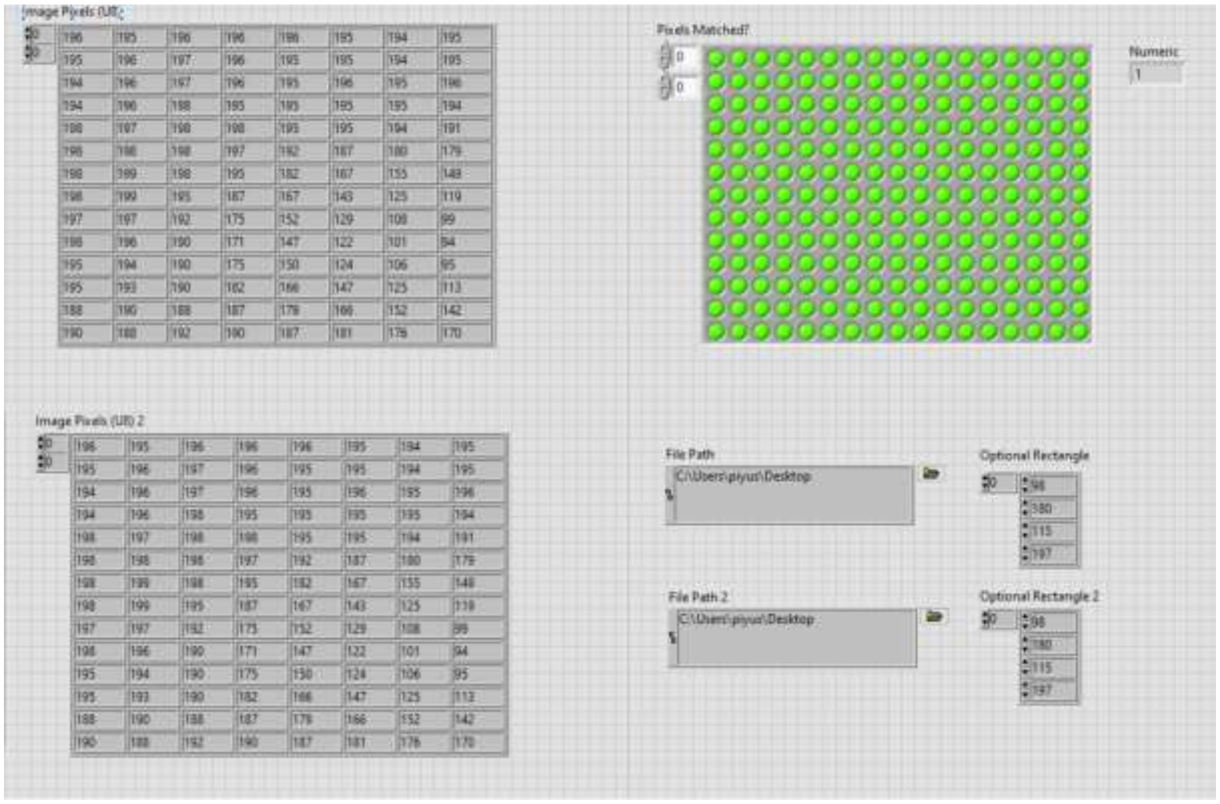


Image Pixels (U8)

196	195	196	196	196	195	194	195
195	196	197	196	195	195	194	195
194	196	197	196	195	196	195	196
194	196	198	195	195	195	195	194
198	197	198	198	195	195	194	191
198	198	198	197	192	187	180	179
198	199	198	195	182	167	155	148
198	199	195	187	167	143	125	119
197	197	192	175	152	129	108	99
198	196	190	171	147	122	101	94
195	194	190	175	150	124	106	95
195	193	190	182	166	147	125	113
188	190	188	187	179	166	152	142
190	188	192	190	187	181	176	170

Image Pixels (U8) 2

196	195	196	196	196	195	194	195
195	196	197	196	195	195	194	195
194	196	197	196	195	196	195	196
194	196	198	195	195	195	195	194
198	197	198	198	195	195	194	191
198	198	198	197	192	187	180	179
198	199	198	195	182	167	155	148
198	199	195	187	167	143	125	119
197	197	192	175	152	129	108	99
198	196	190	171	147	122	101	94
195	194	190	175	150	124	106	95
195	193	190	182	166	147	125	113
188	190	188	187	179	166	152	142
190	188	192	190	187	181	176	170

Pixels Matched? 0 0

Numeric: 1

File Path: C:\Users\piyus\Desktop

Optional Rectangle: 0 — 98, 180, 115, 197

File Path 2: C:\Users\piyus\Desktop

Optional Rectangle 2: 0 — 98, 180, 115, 197