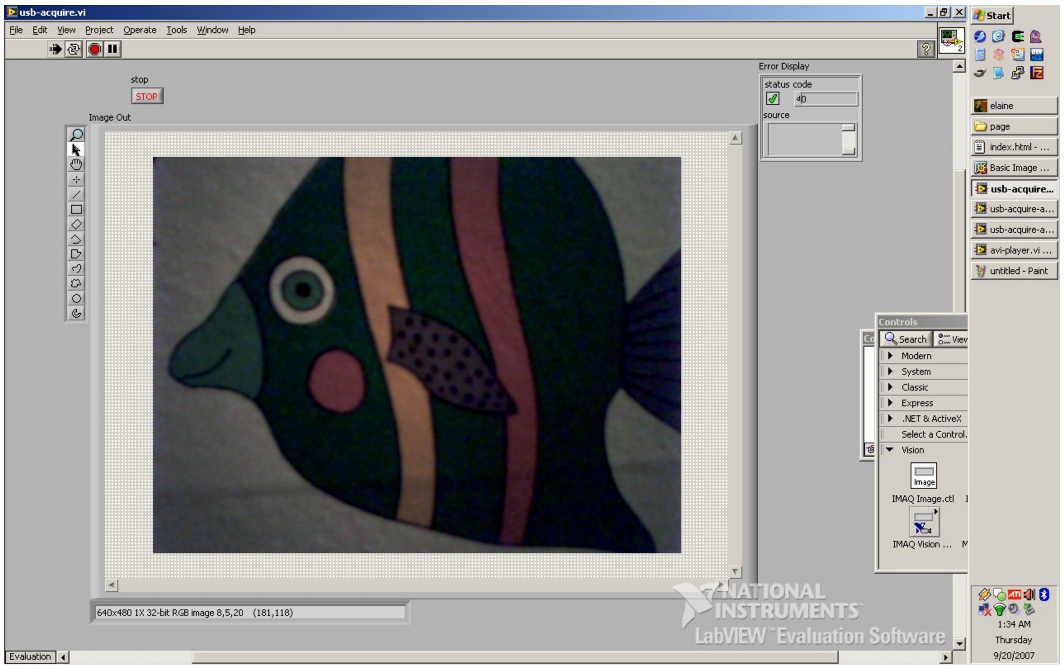
Basic Image Acquisition with NI Tools

Author: Bob Sherbert

Keywords: NI, LabVIEW, IMAQ, Image Acquisition, AVI



Real time image acquisition is a prerequisite to any kind of practical video processing. Simple VIs can be constructed with National Instrument's Vision Development and Image Acquisition packages to facilitate this process.

- Motivation and Audience
- Software Packages Used
- Programming
- Final Words

Motivation and Audience

The first step toward any kind of image processing is the acquisition of the actual images to be processed. The images can be obtained from a camera (of which USB and DV are but two possible varieties) or from a file stored on a local hard disk. Once the images are obtained, they can be further processed by programs to extract whatever information is both desired and discernible from them. Of course, such extraction programs will rarely be perfect on their first execution and, in the course of debugging them, it is often helpful to have a perfectly replicable input. This can help narrow the scope of a search for bugs or errors. In a vision application this replicable input may be accomplished by saving a single stream of data and piping it into subsequent program executions. This article focuses on performing these two critical low level tasks (image acquisition and saving/recall) within the National Instruments LabVIEW framework.

This tutorial assumes that the reader has the following skills/experience:

- Basic to Intermediate LabVIEW programming skills

Software Packages Used

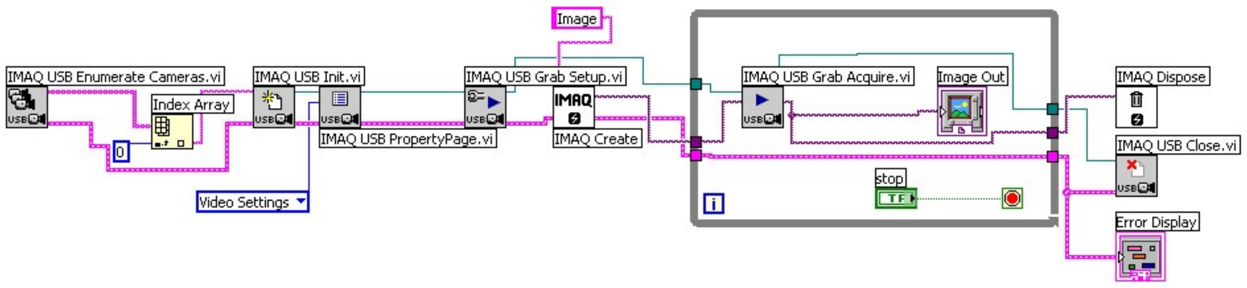
TABLE 1: Parts required for construction

PART DESCRIPTION	VENDOR	PART	PRICE (2007)	QTY	Evaluation Available
LabVIEW 8	NI	776671-09	$600	1	Y
NI Vision Development Module	NI	777859R-09	$824	1	Y
NI Vision Acquisition Software	NI	778413-01	$100	1	Y
NI Vision Acquisition USB Extension	NI	NA	$0	1	NA
Logitech QuickCam Chat (or comparable)	Logitech	961462-0403	$30	1	NA

Programming

LabVIEW
Code is available in the archive basicIMAQ.zip

USB Image Acquisition - usb-acquire.vi

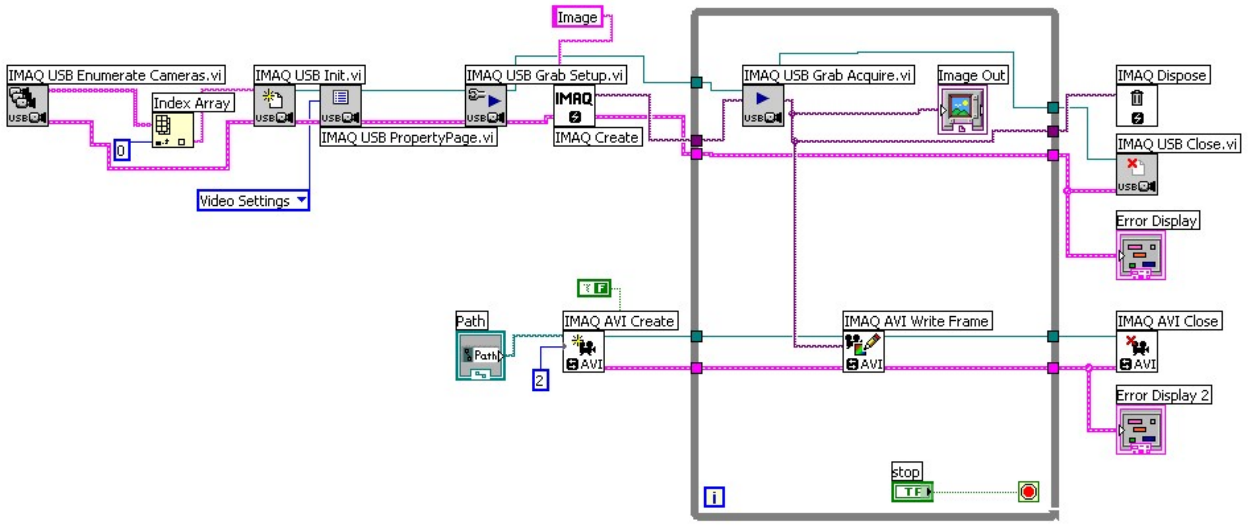


The first step in utilizing images for vision applications is in acquiring the image stream from a camera. This source can be a USB webcam, DV (firewire) camera, etc. Here I have chosen to describe the usage of a USB webcam because the hardware is commonly and cheaply available. Other camera types can be used similarly (from a software point of view) by swapping out the VIs show here for their counterparts for your camera type.

The USB image stream acquisition process is fairly straightforward and can be broken into three steps. The first step is opening the camera and setting up the appropriate data containers for the images, the second is a loop to grab single frames from the camera, and the third is to close the camera and tidy up.

1. **Setup** - First, the 'Enumerate Cameras' VI is used to obtain a list of the devices available on the system. For the sake of simplicity, I have chosen to assume that only one camera is present on the system. It is possible to create a menu to choose between multiple cameras, but this is outside the scope of this article. The identification string for the camera to be use is passed to the 'USB init' VI which takes control of the camera from the system. It passes a resource identifier to the 'PropertyPage' VI which presents the user with a dialog box allowing them to select resolution, frame rate, and other properties of the camera. The 'Grab Setup' VI finishes the preparation of the camera for image acquisition. In addition to preparing the camera, a container must also exist to hold the image data that is going to be pulled from it. In the NI Vision system this takes the form of the IMAQ container (short for IMage AcQuisition). This container must receive a string as input which will be used as the name of the image, and returns an image container.
2. **Grab Frame** - The IMAQ container and camera identifier created in the first step are passed into a loop which calls the 'Grab Acquire' VI. This VI copies an image frame from the camera data stream into the the IMAQ container as well as an image display created to allow the user to see what is going on. Each iteration of the loop overwrites the contents of the IMAQ container, there is no need to create a new one for each iteration.
3. **Clean Up** - When the user terminates the loop the IMAQ container and the camera resource string are both passed to the appropriate disposal functions. An error output block has been added for the sake of assistance in debugging.

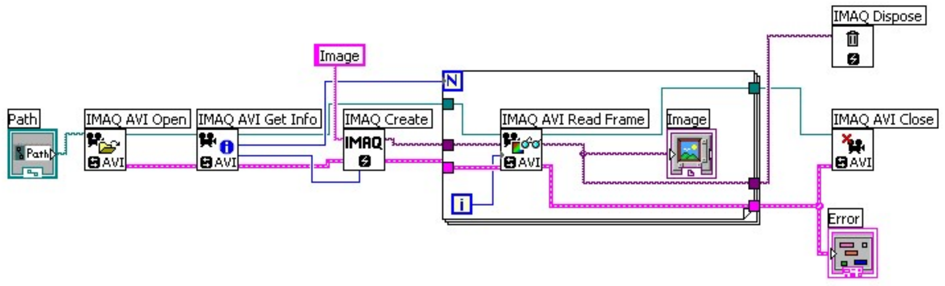
Saving to AVI - usb-acquire-avi.vi



A useful tool for debugging vision applications is the ability to replicate an input over multiple programs runs. This can be done by saving image data to an AVI file and using the file as an input to subsequent executions of the program.

Saving the image data is fairly straightforward. You will notice that this VI, which can be used to save image streams into an AVI file, is very similar to the first example. It differs only in the addition of the AVI file manipulating VIs. At the beginning of the program, while the camera is being opened, an empty AVI file is created using a path specified by the user on the VI's front page. During the main loop, as each frame is pulled from the image stream, it is copied into the AVI file with the 'AVI Write Frame' VI. When the loop is terminated, the AVI is closed along with the camera and the IMAQ container.

Retrieving Streams from AVI - avi-player.vi



For saving image streams as AVIs to be useful for debugging purposes, the user also needs the ability to retrieve these files. Once an AVI is opened, the data from it can be passed into the same IMAQ containers used by the 'USB frame grab' VI, making them nearly interchangeable for usage in larger vision programs. Opening an AVI for image acquisition follows a very similar path to opening a USB camera.

1. **Setup** - A path specified by the user on the VI's front page is used to identify the location of the AVI. From there the 'AVI Open' Function is called to open the file and obtain a reference ID which can be used by subsequent functions. The 'Get Info' VI reports the image characteristics (resolution/color space) to the 'IMAQ create' VI in order to assure that the container is properly sized for the data it will receive. The 'Get Info' function also returns the number of frames in the AVI, which can be used to setup a for loop of appropriate length.
2. **Grab Frame** - The 'AVI Read Frame' VI functions analogously to the 'Grab Acquire' VI from the USB segment. The IMAQ container is passed to it as an input, along with the index of the for loop (telling it which frame to draw out of the file), and the VI returns the appropriate frame as a single image. Which is passed to a display for the user's benefit.
3. **Clean Up** - When all frames have been displayed, the AVI file and the IMAQ container are closed and the program terminates.

Final Words

With the completion of this tutorial the reader should feel confident working with basic NI vision acquisition, image storage and image retrieval with AVI. These skills are essential to developing machine vision applications as they provide the framework for pulling image data into a program and the tools to save, restore, and therefor debug, vision based programs. The programs showed here are very minimalist, showing only the function essential to dealing with image data. They can easily be extended to include robust image processing as your vision application requires.

The author can be reached by email