NI myRIO

Vision Essentials Guide

Ed Doering

POWERED BY LabVIEW

NATIONAL INSTRUMENTS

NI myRIO Vision Essentials Guide

Ed Doering
Electrical and Computer Engineering Department
Rose-Hulman Institute of Technology

Printed July 23, 2015. Download the latest version at `http://www.ni.com/myrio/vision-guide`.

Contents

1 Introduction

Welcome to the *NI myRIO Vision Essentials Guide*, and welcome to the exciting world of machine vision with NI myRIO! Machine vision systems play a critical role in manufacturing automation, shape and color analysis, and robotics. With little more than a USB webcam you can create NI myRIO applications that interact with the visual world to sense motion, take physical measurements, read barcodes and printed labels, inspect products for defects, and respond to colors. Add LCD displays, switches, and servomotors and you can create a complete stand-alone application that controls physical apparatus such as a marble sorter and an auto-panning camera.

This book serves as your guide to the essential techniques necessary to add vision to your NI myRIO project. You will learn through a mixture of written materials, video tutorials, and guided hands-on projects. Part I, *Machine Vision Fundamentals*, introduces you to a generalized application development flow suitable for all of the machine vision projects in this book. The development flow includes defining the application's requirements, configuring the imaging system and acquiring representative images, calibrating to real-world units, developing the vision processing script with NI Vision Assistant, and developing and validating the complete machine application with NI LabVIEW. Part I also introduces you to the "Queued State Machine" design pattern that serves as the basis of the "Machine Vision App" (MVA) LabVIEW project template that you will use to implement each of the application projects. With this template you concentrate on the relevant details for each application (vision script and application-specific LabVIEW coding) instead of getting bogged down trying to create the entire application from scratch.

Part II, *Introductory Machine Vision Projects*, presents four projects to ground you in the basics of image acquisition, stereo vision, NI vision-related software tools, and application development with the "Machine Vision App" project template. These introductory projects include detailed step-by-step in-

structions to help you quickly get up to speed before tackling the more advanced projects of Part III, *Machine Vision Application Projects*. Each of these ten design-oriented projects presents a set of functional requirements and then offers a recommended technical approach to develop the vision script and LabVIEW implementation based on the MVA template project. Each of these open-ended design projects introduces you to new machine vision concepts and associated NI Vision implementation techniques. Once you complete these projects you will be able to create sophisticated embedded vision projects of your own imagining!

Intended Audience

This book is intended for students at the senior year and early graduate levels, especially those students engaged in capstone projects and research. A background in electrical and computer engineering is helpful, but students pursuing other disciplines will find the level of tutorial detail to be more than adequate.

Book Resources

The most-recent version of *NI myRIO Vision Essentials Guide* is always available as a free download at `http://www.ni.com/myrio/vision-guide`; page iv shows the document version. The download also includes the "Machine Vision App" template LabVIEW project that serves as the starting point for each of the hands-on projects.

This document is fully hyperlinked for section and figure references, and all video links are live hyperlinks. Open the PDF version of this document for the most efficient way to access all of the links; click a video hyperlink to automatically launch the video in your browser. Within the PDF, use `ALT+leftarrow` to navigate back to a starting point.

Part IV, *Appendices*, provides the following supplements:

- Appendix A, *Recommended Equipment* – details each piece of equipment used to implement the lab projects including product page links,
- Appendix B, *MXP and MSP Connector Diagrams* – shows the NI myRIO I/O resources and connector pin numbers,
- Appendix C, *LabVIEW Quick Tips* – summarizes commonly-used LabVIEW keyboard shortcuts to promote coding efficiency,
- Appendix D, *Build a Stand-Alone Application* – learn how to deploy the LabVIEW project as a stand-alone application on the NI myRIO so that it runs automatically after myRIO powers up,

- Appendix E, *Marble Sorter Construction Details* – presents the hardware apparatus that serves as the basis of the *Marble Sorter* application project (Chapter 17), and
- Appendix F, *Video Tutorial Links* – summarizes all of the video links and groups them into related categories for easy reference.

For the Instructor

All of the projects in this book are designed to be graded activities. The "Machine Vision App" template LabVIEW project includes an "images" subfolder where students can place the representative images that they used to develop their vision script and a "documentation" subfolder in which students can place a lab report and any other documentation that you wish. The project folders may then be bundled into a single .zip file and uploaded to your course management system for grading, and you will have electronic access to every file related to the student's project. Should you prefer a paper-based approach to grading, request the students to print the "Main.vi" front panel diagram and all hidden subdiagrams (see the LabVIEW "File | Print..." menu item with the "VI documentation" option selected) as well as the front panel of the data highway cluster type definition and you will be able to see the complete design.

A solutions manual is available to qualified instructors; please contact your local National Instruments representative for details.

1.1 Acknowledgements

This book represents the contributions of many individuals, and without them this book project would not have been possible. I gratefully acknowledge Gretchen Edelmon for her technical support, advice, and encouragement regarding the book's content and organization; Thomas J. Bress for his excellent textbook *Effective LabVIEW Programming*[1] that introduced me to the Queued State Machine design pattern that serves as the basis of the "Machine Vision App" project template; Isaac Sanchez and Alex Drane for timely advice on LabVIEW vision techniques; and Margaret Barett for inviting me to participate in the original NI myRIO Beta Program. I am also grateful to Andrew Watchorn for his ongoing assistance and to Tom Robbins at NTS Press for his editorial support on this project.

Ed Doering
Department of Electrical and Computer Engineering
Rose-Hulman Institute of Technology
Terre Haute, IN 47803

doering@rose-hulman.edu

[1] http://www.ntspress.com/publications/effective-labview-programming, NTS Press, 2013.

Part I

Machine Vision Fundamentals

2 Application Development Flow

The machine vision projects in this manual generally follow a well-defined *application development flow* from original concept through validation of the application on the NI myRIO target. The particular details vary considerably from one project to the next, but the overall development flow remains the same.

Familiarize yourself with this development flow, and refer back to this chapter periodically as you develop your own machine vision application.

> **NOTE**: *The application development flow described here may also be used to create a desktop application in case you do not have access to the NI myRIO target.*

Figure 2.1 on the following page visualizes the application development flow as an eleven-step procedure, and the remainder of this chapter lists important considerations for each step:

1. Define the requirements of the machine vision application
    - Processing objectives, e.g., measurement and gauging, classification and sorting, and bar code recognition
    - Throughput measured in parts processed per second
    - Minimum feature size and number of pixels allocated to this feature
    - Size of the field of view
    - I/O devices for part sensing, position control, and information display
2. Configure the image acquisition system
    - Determine the minimum feature size in physical units such as millimeters

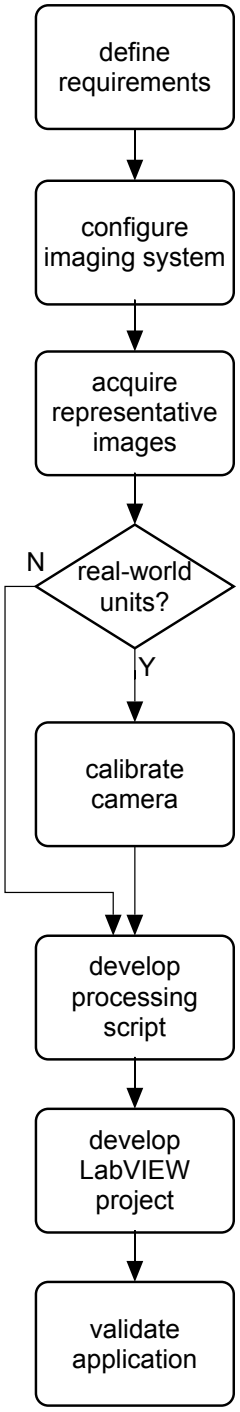

Figure 2.1: Development flow for a machine vision application.

- Determine the number of pixels to allocate to the minimum-sized feature, and thereby obtain the required spatial resolution in millimeters per pixel
- Select the camera resolution and camera-to-object distance to achieve the required spatial resolution, field of view (FOV), and frame rate
- Arrange the lighting system to obtain high contrast for the features of interest and to suppress distracting background features that complicate subsequent software processing and add unnecessary computational cost
- Use grayscale images for efficiency unless color is required for the application; color images tend to slow down the display, especially at higher resolutions

3. Acquire representative images

   - Connect the camera to the desktop or to the NI myRIO
   - Use Vision Assistant to collect images over the range of possible conditions: Select the "Target" as "This Computer" for a desktop-connected camera and as "Select Network Target" when the camera is connected to myRIO; use IP address `172.22.11.2` when myRIO is tethered by its USB cable otherwise use the IP address assigned to myRIO by the wireless network.
   - Use manual camera settings, especially focus; record the settings for later use
   - Collect an image of one or more calibration targets using the same camera settings

4. Calibrate the camera when the application requires real-world units

   - Use the calibration target image(s) collected in the previous step
   - Use the simplest calibration technique that achieves the required measurement accuracy
   - Do not disturb the image acquisition setup once calibrated

5. Develop the processing script

   - Use the stand-alone version of NI Vision Assistant
   - Begin with the color-to-luminance step (unless the application requires color, of course)
   - Evaluate the performance using the representative images collected earlier

- Use the minimum number of steps that satisfies the requirements
- Periodically evaluate the computational effort of each step with the Vision Assistant "Performance Meter"

6. Create a new NI myRIO LabVIEW project
   - Connect the camera to NI myRIO if not already connected
7. Configure the Vision Acquisition Express VI
   - Select the "Continuous acquisition with in-line processing" option
   - Adjust the camera video mode and attributes to match the settings recorded in Step 3
   - Click "Test" to confirm that the image looks correct
   - Enable the frame rate output, if desired
8. Configure the Vision Assistant Express VI
   - Open the Vision Assistant script created in Step 5
   - Remove the color-to-luminance step (assuming the camera is in grayscale output mode)
   - Click through the steps to confirm correct operation
   - Select the necessary input controls and output indicators
   - Enable the "Create Destination Image" option
   - Move the Express VI inside the while-loop structure
   - Create the needed controls and indicators
9. Add block diagram code for I/O devices
   - Use available on-board devices as needed, e.g., LEDs, accelerometer, PWM output for servomotors, and switches
   - Refer to the *NI myRIO Project Essentials Guide* for detailed instructions on a wide range of useful external devices such as proximity sensors and LCD display
10. Validate the finished application
11. Deploy the VI as a stand-alone application
    - Refer to Appendix D on page 141 for complete details.

3 Design Pattern for Machine Vision Applications

A LabVIEW *design pattern* is a particular block diagram coding style that has emerged over time and has been "vetted" by many developers as a useful way to develop a broad range of specific applications. Design patterns range from the very simple to the very complex, but all design patterns provide a systematic way to transform your concept into a functional and reliable software application.

This chapter introduces the popular *state machine* design pattern, especially the "Queued State Machine" that serves as the foundation of the "Machine Vision App" template LabVIEW project from which all of the projects in this lab manual may be developed.

3.1 State Machines

The excellent textbook *Effective LabVIEW Programming*[1] by Thomas J. Bress presents a wide range of design patterns based on the *state machine*. A *state* embodies a specific action such as incrementing a counter value, fetching the next frame from the webcam, analyzing the image to search for a barcode, saving an image to disk, and so on. The state "machine" moves from one state to the next in either an unconditional fashion (State C always follows State B) or conditionally (go to State C if a particular front-panel button is pressed, otherwise go to State D). See *State Diagrams*[2] for a more complete discussion of state diagrams.

This "classic state machine" design pattern uses a while-loop, a shift register

[1] `http://www.ntspress.com/publications/effective-labview-programming`, NTS Press, 2013.

[2] `http://youtu.be/n-wJjQKCEeo` (10:06)

to maintain the current state, and a case structure to implement the activities for each state and also to determine the next value of the state register. On each while-loop iteration the value of the state register is read to select a particular subdiagram of the case structure, therefore, each subdiagram embodies a single state. Each subdiagram calculates the next value of the state register which is then fed back via the while-loop's shift register to become the next value of the state register. Refer to *Classic State Machine*[3] for a more complete discussion of the LabVIEW implementation of a classic state machine.

3.2 Queued State Machine

The "Queued State Machine" (QSM) is a more sophisticated approach that decouples a state's activity from the mechanism to determine the next state. This decoupling makes the QSM much easier to develop and maintain because it promotes a "divide and conquer" approach to breaking up the application design into manageable and well-defined activities. Each state can be developed and debugged independently of all the other states.

But how is the next state determined? A special state called the *scheduler* takes care of this. The scheduler examines image analysis results, front-panel buttons, and external sensors to determine which *task* — an ordered sequence of states — should be performed next. The scheduler applies this ordered sequence of states comprising a task to the input of a *queue*, also known as a FIFO (first-in, first-out) memory structure. Once *enqueued* the states are now ready to be read out one at a time, i.e., *dequeued*, on each while-loop iteration to select the associated case-structure subdiagram activities in the proper sequence. The queue serves the same role as the state register in the classic state machine. After enqueueing a single task, the scheduler must finish by enqueueing itself so that that state machine will return to the scheduler state after completing the current task.

Figure 3.1 on page 19 illustrates the task diagram for a generic QSM intended for machine vision applications that target NI myRIO. Each box corresponds to a state, and a bundle of states indicates a task. The QSM executes the start-up task and then schedules tasks according to conditions determined by analysis results, front-panel buttons, and external controls and sensors. The scheduler typically selects the default task when all other task conditions evaluate to "false." The default task retrieves an image from the webcam, analyzes the image, adds a nondestructive overlay, and reads and writes NI myRIO on-board and external devices. When the front-panel Stop button is pressed or an error condition is detected the scheduler selects the shutdown task to halt the application.

[3] `http://youtu.be/muudWDUkypM` (6:53)

3.3 "Machine Vision App" Template

Download the .zip file from the *NI myRIO Vision Essentials Guide*[4] main page, extract the "Machine Vision App" folder, open the "machine vision app" LabVIEW project (double-click the .lvproj file), and then double-click Main.vi to open its block diagram. Take a few moments to study the structure of the block diagram, and especially scroll through the case-structure subdiagrams to see the various states and their associated activities.

This template provides a single starting point for each of the projects in this lab manual. As you complete each project and develop your own library of machine vision applications you may find it easier to work from a copy of an existing project that is similar to the requirements of the new project. In either case, implementing a new machine vision application will become a relatively straightforward task once you understand the basic principles of the queued state machine design pattern.

> **IMPORTANT**: *Do not be discouraged by the seeming complexity of the queued state machine template. You will soon discover that the template is easy to edit and achieves expected and reliable behavior without expending unnecessary development effort.*

QSM Structure

Figure 3.2 on page 20 shows the task diagram for the "Machine Vision App" (MVA) template project. The task diagram shows only two tasks but the template includes many empty states that you can adapt to the needs of your project. The default task applies an edge detector to the webcam image and displays either this processed image or the original webcam image along with a nondestructive graphics overlay that shows the system date and time. Figure 3.3 on page 21 shows the state sequence for the default task.

Pressing the front-panel save image *action button* selects the ***Save Image*** task (Figure 3.4 on page 22); this task saves the displayed image to an image file on the attached USB flash drive. The action button remains pressed until released, therefore the task releases the button as its final action. Because all tasks that respond to action buttons must conclude in a like manner, the button release behavior is implemented in a single state that can be included in any task.

Figure 3.5 on page 23 shows the state sequence for the start-up task. These states initialize data values, allocate memory for image buffers, reset front-panel buttons, and configure on-board and external devices. Figure 3.6 on page 24

[4] http://www.ni.com/myrio/vision-guide

shows the state sequence for the shutdown task. These states stop the webcam, perform general "housekeeping" to close log files and deallocate memory, and return peripheral devices to inactive states.

Template Walk-Through Tutorials

Take the time to carefully study each of the following video tutorials. Once you have completed this study you will understand all of the essential techniques to edit and adapt the template project to the requirements of your project:

Develop your "big picture" understanding of the Queued State Machine design pattern:

1. *MVA Project Files and Folders*[5] – Open the "machine vision app" folder, review the files and folders, open the .lvproj file, and view the organized files.
2. *MVA Main.vi Quick Tour*[6] – Quick-paced tour of the "Main" VI: while-loop, case-structure, state queue, front-panel controls, NI myRIO ports, and data highway.
3. *MVA States*[7] – Review the standard states (qsm, myRIO, image), the example application-specific state, and the expansion states.
4. *Run the MVA Project*[8] – Open the "machine vision app" folder, open the .lvproj file, open Main.vi, run the VI, demo the buttons, and stop. NOTE: The webcam must already be configured!
5. *"Initialize Queue" VI*[9] – Walk through the "initialize_queue" VI.
6. *"Enqueue States" VI*[10] – Walk through the "enqueue_states" VI.
7. *"Dequeue State" VI*[11] – Walk through the "dequeue_state" VI.

Template Editing Tutorials

These detail-oriented tutorials illustrate all of the essential techniques necessary to adapt the template to your needs:

[5] `http://youtu.be/AekhpmU9s3Y` (2:04)
[6] `http://youtu.be/sgGgU2dkocI` (4:52)
[7] `http://youtu.be/IKFeOF4Zh6Y` (2:18)
[8] `http://youtu.be/MITs9SQz-Pg` (1:38)
[9] `http://youtu.be/p97xHbXmMgg` (1:20)
[10] `http://youtu.be/AFg-_tv9MVU` (2:16)
[11] `http://youtu.be/-uIB_5KlstQ` (2:30)

Data highway: The *data highway* is a combination of a LabVIEW *cluster* and a while-loop shift register. The cluster is a collection of front-panel controls stored as a "type def" (type definition); any change to the type definition propagates throughout the entire project. A cluster constant initializes the while-loop shift-register and defines the data highway "lanes." Because the data highway passes through the case structure subdiagrams, each QSM state can read and write values that in turn can be accessed by every other state.

1. *Change Data Lane Name*[12] – Change a data highway lane name: edit the type def, and then observe the change in the bundle/unbundle functions.
2. *Add Data Lane*[13] – Create a new data highway lane: edit the type def, bundle and merge, and access in another state.
3. *Remove Data Lane*[14] – Remove an unused data highway lane: edit the type def, delete the control, remove the element in the cluster bundle.
4. *Add Custom Data Lane*[15] – Add a custom lane: create a control from the Vision Assistant Express VI indicator output, convert to type def, save type def as a control, add the control to the data highway, merge the Vision Assistant Express VI output onto the data highway, and then retrieve the value in another state.

"state" type def: The state enumerated data type control selects the subdiagram in the case structure. state is a type definition, therefore any modifications propagate throughout the project. Changing a state name also changes the name in the case-structure selector. The state and the case-structure are closely related, therefore these videos show how to edit both at the same time.

1. *MVA States*[16] – Review the standard states (qsm, myRIO, image), the example application-specific state, and the expansion states.
2. *Change State Name*[17] – Change a state name: edit the subdiagram label, open the state type def, edit the enumerated datatype control, observe the changed name in the case-structure subdiagram.
3. *Rearrange States*[18] – Reposition a state in both the state type def and in the case structure's subdiagram ordering.

[12] `http://youtu.be/q2iusxCE3Yo` (0:54)
[13] `http://youtu.be/7UFQNZJDYYU` (1:27)
[14] `http://youtu.be/Kt6aeX3d4Yg` (0:47)
[15] `http://youtu.be/GkqjZCj3xck` (2:58)
[16] `http://youtu.be/IKFeOF4Zh6Y` (2:18)
[17] `http://youtu.be/pj_aboGCvjI` (1:09)
[18] `http://youtu.be/z6wO6RQy3QY` (1:25)

4. *Add State at End*[19] – Add a state at the end: open the state type def, edit the enumerated control, show two ways to create a new case-structure subdiagram.
5. *Remove State*[20] – Remove an unused state in a task, in the case structure, and in the state type def.
6. *Remove State with Review/Accept Changes*[21] – Remove an unused state; "Review and Accept Changes" message.

QSM states: The QSM states perform activities required by all applications. These tutorials discuss each state and typical modifications.

> **IMPORTANT**: *You will often need to edit the activities within the QSM states but you should* **not** *remove them!*

1. ***(qsm) initialize*** – Perform one-time initialization activities such as allocating memory for additional image buffers with IMAQ Create (see *(qsm) initialize: Image Buffer*[22]) and resetting front-panel indicators with Invoke Node (see *(qsm) initialize: Front-Panel*[23]).
2. ***(qsm) release*** – *(qsm) release*[24] – Use an Invoke Node to release front-panel action buttons. This state is normally enqueued at the end of a task that responds to a front-panel action button to automatically release the button; without this state the action button remains depressed and repeatedly triggers the task until the button is manually pressed again.
3. ***(qsm) clean up*** – *(qsm) cleanup*[25] – Perform clean-up activities such as freeing memory allocated to image buffers with IMAQ Dispose.
4. ***(qsm) shut down*** – *(qsm) shutdown*[26] – Release the state queue (frees up its memory) and generate a Boolean "true" to stop the while-loop structure. The VI will terminate execution after this state.
5. ***(qsm) schedule*** – *(qsm) schedule*[27] – Select a single task (an array of states) to be applied to enqueue states VI; a *selector tree* based on the Select node (a 2-to-1 multiplexer) is normally the most convenient method,

[19] http://youtu.be/jJlEtD-2zgU (1:49)
[20] http://youtu.be/F8kjxt0mHTI (1:23)
[21] http://youtu.be/G_k7aLgglgE (2:03)
[22] http://youtu.be/suCkeB_afOc (2:02)
[23] http://youtu.be/eBtBy__PNwU (1:49)
[24] http://youtu.be/NN4qyYiG7CM (2:13)
[25] http://youtu.be/yA9TaFt6lYQ (1:22)
[26] http://youtu.be/YgLynyjlPo8 (1:30)
[27] http://youtu.be/uS2QlWo3VM4 (4:27)

although case structures could also be used. The selector tree uses front-panel controls and data highway flags to choose the required task. Some tasks are performed once in response to action buttons and conditions. The default task handles ongoing activities related to the image stream and NI myRIO port activity.

myRIO states: The myRIO states perform activities required by applications targeted to NI myRIO. These tutorials discuss each state and typical modifications.

1. ***(myRIO) initialize*** – *(myRIO) initialize*[28] – Perform one-time initialization activities such as opening PWM channels, configuring serial SPI and I^2C-bus ports, and sending configuration strings to external peripherals such as LCD displays.
2. ***(myRIO) clean up*** – *(myRIO) cleanup*[29] – Perform housekeeping activities such as closing PWM channels, returning myRIO on-board devices such as digital and analog outputs to an inactive state, and sending shutdown strings to external peripheral devices.
3. ***(myRIO) update ports*** – *(myRIO) update ports*[30] – Read and write myRIO on-board and peripheral devices. You will often find convenient or necessary to place the myRIO Express VIs in other states, too, but ***(myRIO) update ports*** is a useful starting point.

image states: The image-related states perform standard activities related to obtaining, analyzing, and displaying images, including images with nondestructive information overlays.

1. ***(image) get*** – *(image) get*[31] – Use the Vision Acquisition Express VI to grab the next webcam image and store it in the webcam buffer, and optionally use IMAQ Symmetry to flip the image right side up when using the tabletop camera copy stand. The Express VI allocates memory to the image buffer only on its first run. Note that the Express VI output is a *reference* (pointer) to the webcam buffer and does *not* contain any image data itself. This reference must be merged onto the data highway so that other states can find the webcam image. See *(image) get: Vision Acquisition*[32] to learn

[28] `http://youtu.be/KtGDT-jTtmg` (1:40)
[29] `http://youtu.be/fi1GUcoloj0` (1:12)
[30] `http://youtu.be/3gNSpF4RnXA` (1:49)
[31] `http://youtu.be/guqZisxPUGw` (2:23)
[32] `http://youtu.be/jrhG9sOwGIQ` (3:13)

how to use NI-MAX while configuring Vision Acquisition to select the webcam resolution (mode) and attributes.

2. ***(image) analyze*** – *(image) analyze*[33] – Use the Vision Assistant Express VI to load the script created during development with the stand-alone Vision Assistant tool, select the necessary options to create an image buffer for the analyzed image (allocates memory only on the first run), and then select the necessary controls and indicators so that the script inputs and outputs can be merged into the data highway. The Express VI requires a reference to the webcam image buffer as its input and generates a reference to the analyzed image buffer as its output.

   Most Vision Assistant script steps produce clusters or arrays that are not available as standard front-panel controls when editing the data highway, however, it is very easy to create a custom control that can be added to the data highway; see *Add Custom Data Lane*[34] for details.

3. ***(image) overlay*** – *(image) overlay*[35] – use IMAQ Overlay Text to add a nondestructive information overlay to an image buffer. This is a relatively simple example, but is illustrative of the general technique to add overlays. LabVIEW provides a wide variety of VIs to create overlays.

4. ***(image) show*** – *(image) show*[36] – update the main image display using any of the available image buffers; the Select node and a mode button are illustrated here.

[33] `http://youtu.be/mKuwsgGs7sY` (2:23)
[34] `http://youtu.be/GkqjZCj3xck` (2:58)
[35] `http://youtu.be/Vmony2qtnH8` (3:12)
[36] `http://youtu.be/LrjuylUE_4Q` (1:31)

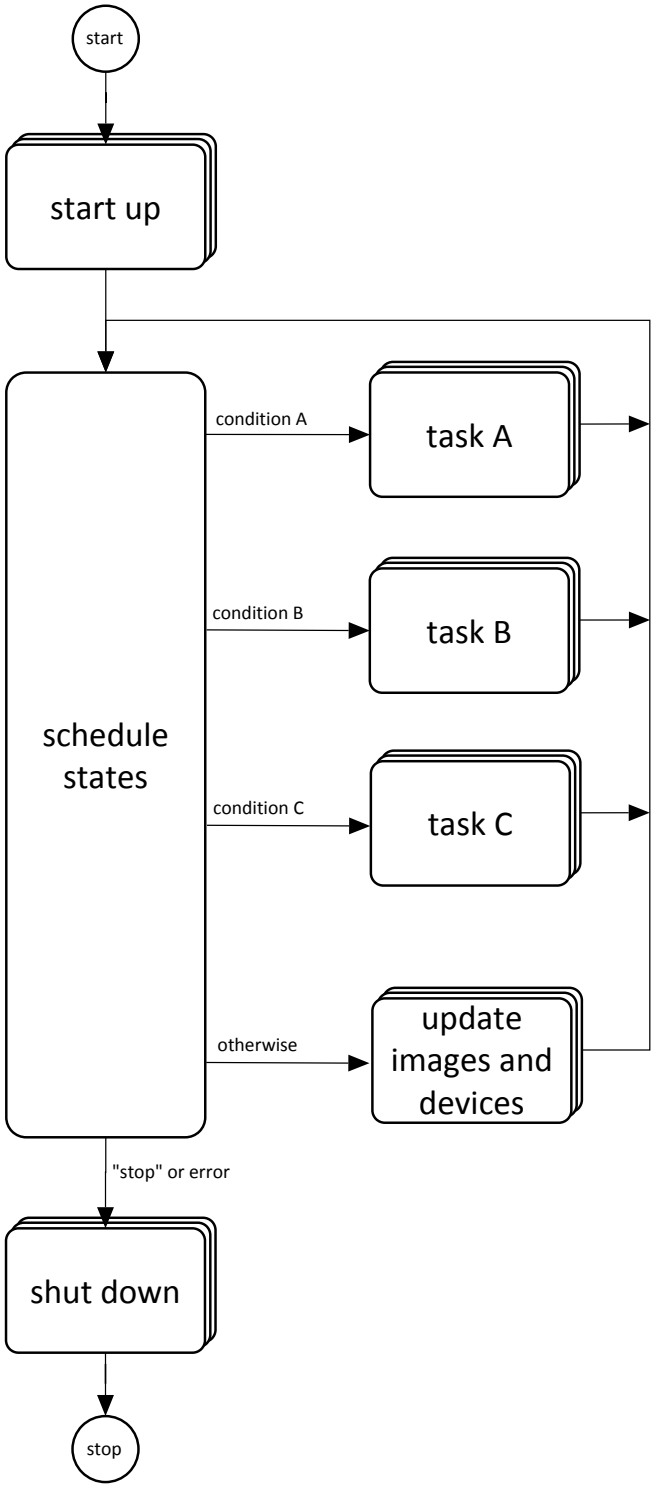

Figure 3.1: Task diagram for the Queued State Machine (QSM).

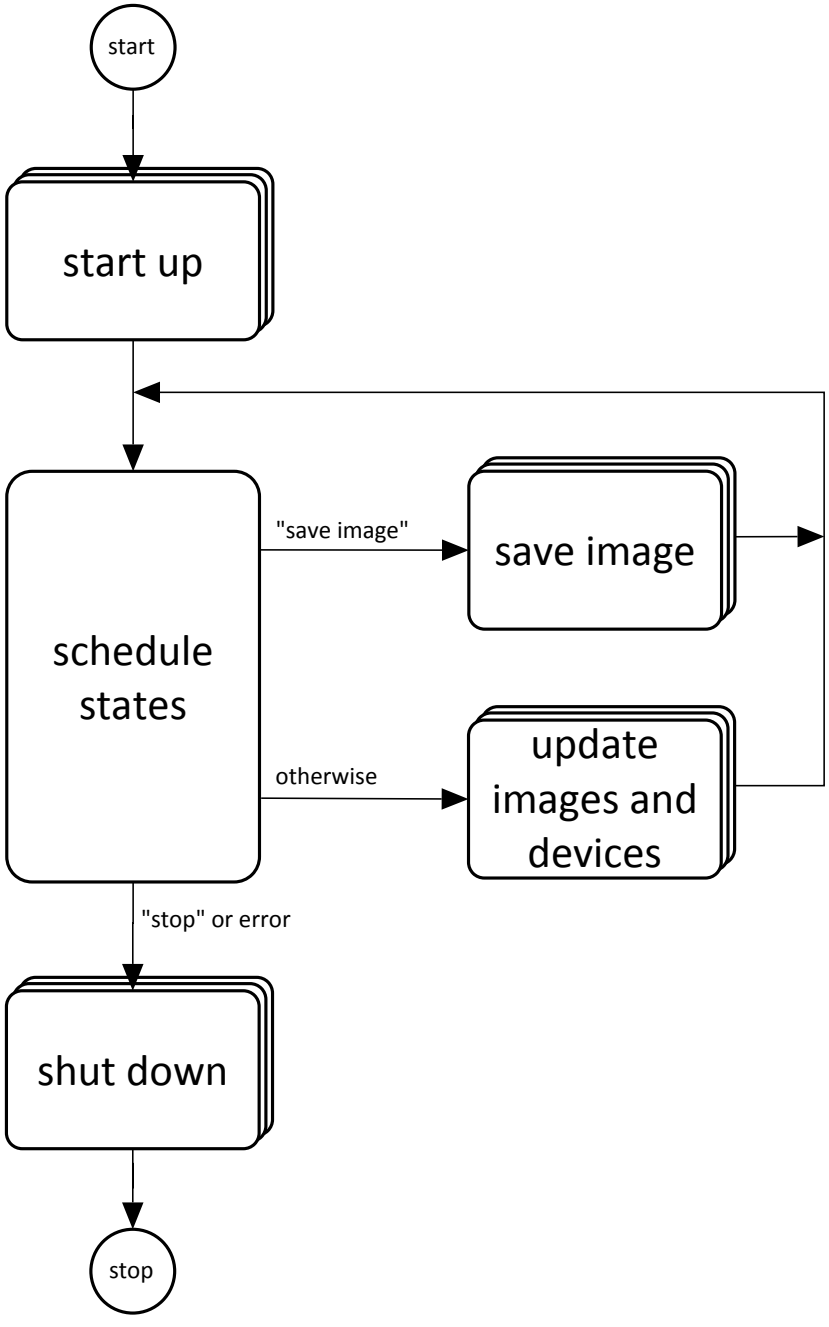

Figure 3.2: Task diagram for the “Machine Vision App” (MVA).

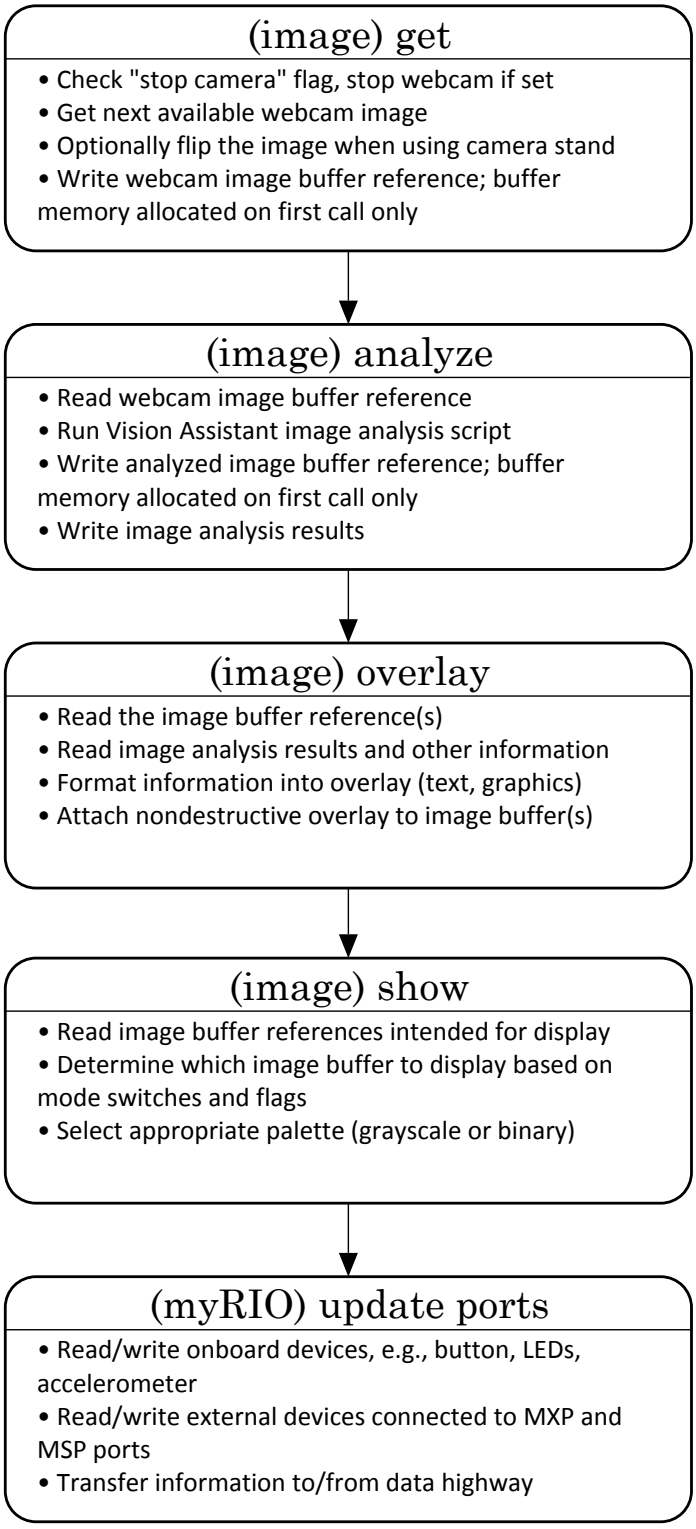

Figure 3.3: State sequence and activities for the MVA default task.

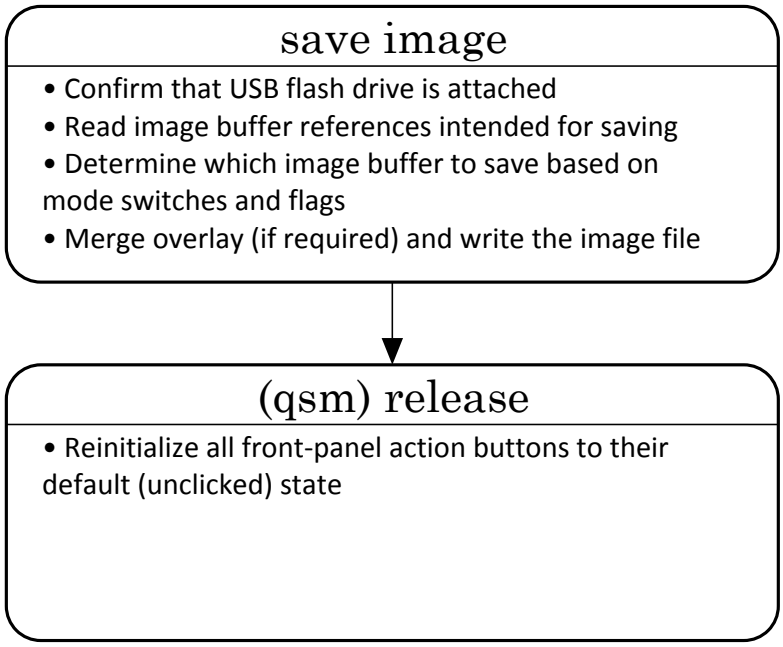

Figure 3.4: State sequence and activities for the MVA "save image" task.

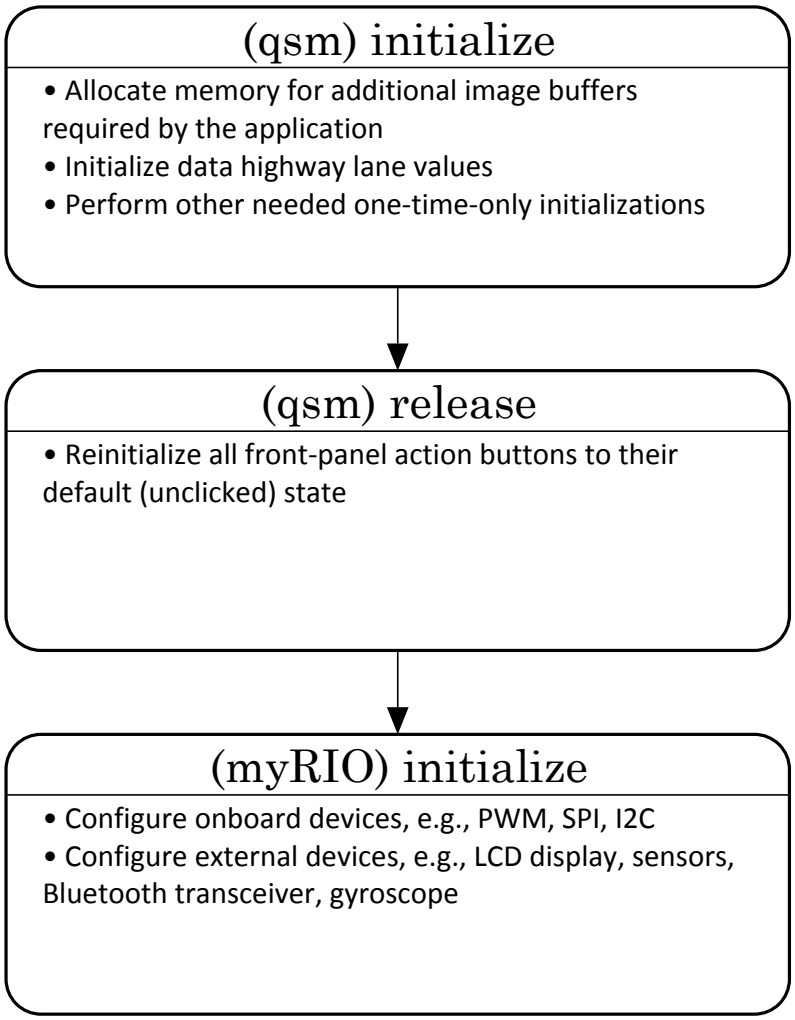

Figure 3.5: State sequence and activities for the MVA start-up task.

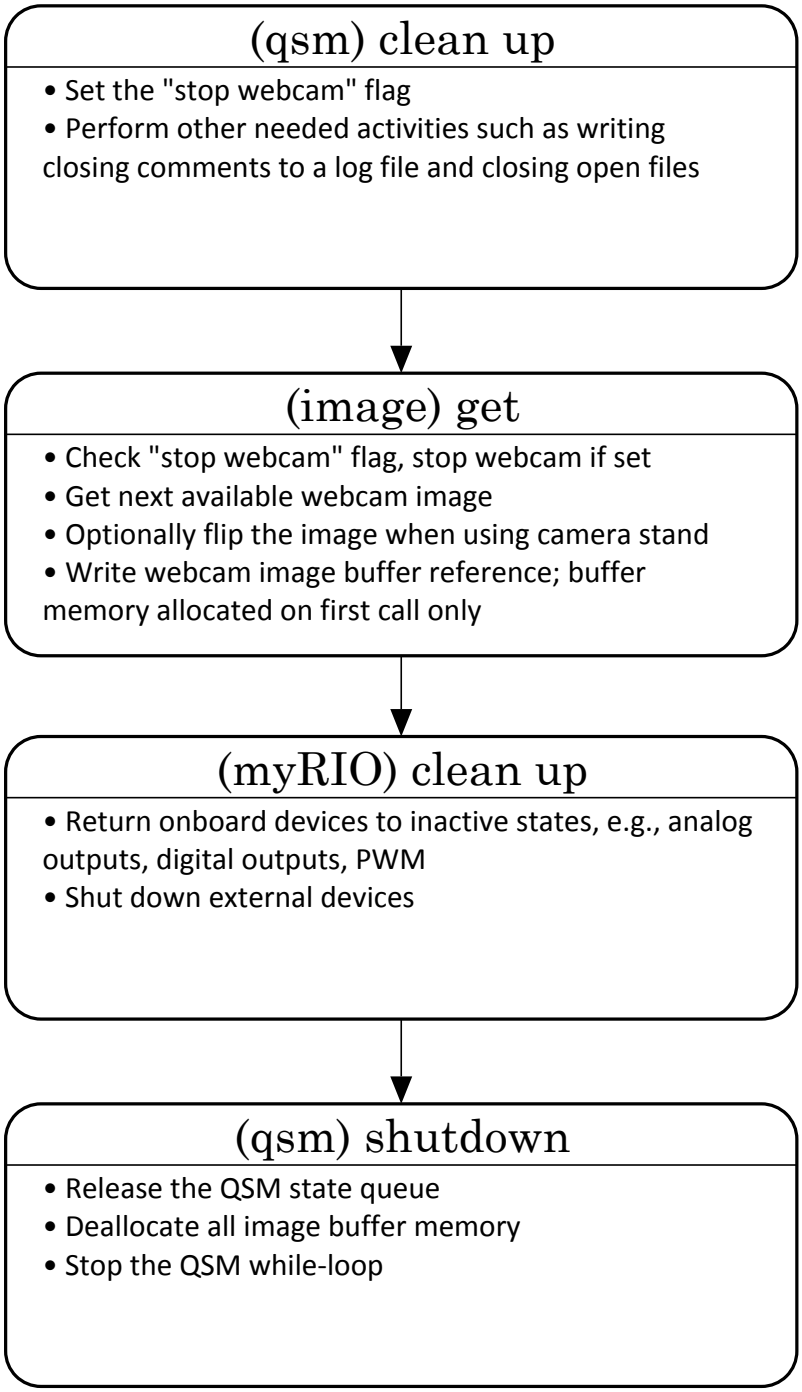

Figure 3.6: State sequence and activities for the MVA shutdown task.

Part II

Introductory Machine Vision Projects

4 Camera Setup

4.1 Synopsis

Camera setup constitutes an important early step in the development of every machine vision application. Everything hinges on the *minimum feature size* and the number of pixels to allocate to this size; these two values define the required image spatial resolution which in turn can be adjusted by a suitable camera-to-object distance. Camera setup also involves *calibration* to allow the machine vision application to make physical measurements in *real-world units* instead of pixels. Calibration can also correct, or at least mitigate, various non ideal effects such a misaligned image sensor and lens distortion.

In this project you will learn how to set up your camera to acquire calibrated images with a desired spatial resolution and field of view.

4.2 Objectives

1. Measure a camera's pixel aspect ratio
2. Determine the camera-to-object distance to achieve a desired image spatial resolution
3. Calculate the field of view (FOV)
4. Calibrate a camera to use real-world units
5. Correct lens distortion and tangential distortion

4.3 Deliverables

1. Screen captures of NI Vision Assistant script and individual step dialog boxes

2. Lab report or notebook formatted according to instructor's requirements

4.4 Required Resources

1. NI Vision Assistant
2. Ruler with millimeter and centimeter marks

4.5 Preparation

Spatial resolution describes the distance spanned by two adjacent pixels. The fixed spatial resolution of the camera's image sensor maps onto the object plane with a variable image spatial resolution R_{S} defined by the camera lens and distance from the object plane. The available *field of view*, or FOV, is likewise determined by the camera distance. Establishing the correct spatial resolution constitutes an important consideration for every machine vision application. Begin with the minimum feature size S_{F} (in physical units such as millimeters) that must be imaged with at least N_{F} pixels. Divide S_{F} by N_{F} to obtain the required spatial resolution R_{S}.

► Study the video *Camera Setup Principles*[1] to learn how to use the minimum feature size S_{F} and the required number of pixels N_{F} to determine the spatial resolution R_{S} and consequently the necessary camera distance and resulting FOV.

► Study the video *Acquire Image in NI Vision Assistant*[2] to learn how to acquire a webcam image, select the image resolution, adjust the webcam attributes such as focus and contrast, and measure the distance between two features.

> **TIP**: *The NI Vision Assistant "Acquire Images" tab defaults to "This Computer" as the target device and displays a list of all desktop-connected cameras from which you can choose. You can also access cameras connected to NI myRIO within Vision Assistant: Choose "Select Network Target" instead of "This Computer" and then enter the IP address as either* `172.22.11.2` *when myRIO is connected by its USB cable or the IP address associated with myRIO by the wireless network.*

[1] `http://youtu.be/ul5sme31cMo` (9:16)
[2] `http://youtu.be/wTJr8beDtvQ` (3:58)

4.6 Pixel Aspect Ratio

► Set up your camera to image a metric ruler with a field of view (FOV) of between 10 and 20 centimeters. Place the ruler in the horizontal direction. Use the horizontal edge of another application's window on your computer desktop to serve as an alignment aid so that the ruler marks follow a single image pixel row.

$\square_1$ Measure the horizontal image spatial resolution $R_{\mathrm{S_H}}$.

$\square_2$ Orient the ruler in the vertical direction and then measure the vertical image spatial resolution $R_{\mathrm{S_V}}$.

$\square_3$ Calculate the ratio of the two spatial resolutions. How close (in terms of percentage) is this *pixel aspect ratio* to unity? In other words, to what degree does your camera's image sensor appear to have square pixels?

4.7 Spatial Resolution, Camera Distance, and FOV

$\square_4$ Measure and tabulate the image spatial resolution for at least four camera-to-object distances. Use limiting values such as the closest distance that still maintains sharp focus and the farthest distance at the limits of your camera copy stand as well as several in-between values. State the external reference mark that you used on the camera. Calculate the camera constant K and the distance d between the reference mark and the projection center. Confirm that the calculated value of d is reasonable by estimating the position of the camera lens with respect to your reference mark.

$\square_5$ Plot the camera distance and the two FOV components as a function of spatial resolution for your camera. Include plot markers to show your measurement points. Save this plot as a convenient way to set up your camera system for other projects. State the image sensor array dimensions N_{H} and N_{V} for your camera.

$\square_6$ Set up the camera to image the (horizontal) ruler so that 1 cm contains 80 pixels. Calculate the image spatial resolution R_{S} in millimeters per pixel, the necessary camera-to-object distance D, and the horizontal field-of-view component FOV_{H}.

$\square_7$ Measure the actual image spatial resolution and compare it to your target value with a percent error calculation. How well does the actual value match your intended value?

$\square_8$ Adjust the ruler position so that the 0 cm mark appears at the left edge of the image; the right edge of the image now displays the horizontal FOV. Record this value, numerically compare to your target value, and comment on the degree of agreement.

4.8 Real-World Units

Up to this point you have manually translated measurements in pixel units to physical units based on your recorded spatial resolution. You can easily *calibrate* the camera so that measurements appear directly in any desired physical or *real-world* units. NI Vision Assistant provides a number of camera calibration tools of increasing sophistication to meet the needs of your particular machine vision application, and you will investigate two of them in this section.

Point distance calibration

The *point distance calibration* is the simplest technique and requires negligible computational effort. This calibration assumes that the image sensor plane is perfectly aligned with the object plane (zero *tangential distortion*) and that the lens is likewise perfectly free of distortion.

► Study the video *Step 2: Calibrate the Camera*[3] to learn how to select two points on a ruler to calibrate the camera.

$\square_9$ Use the same method described in the video to calibrate your camera based on the image of a horizontal ruler with the 0 mm mark visible at the left-hand side of the image. Estimate the maximum error between the calibrated ruler overlay and the ruler itself, and screen capture the ruler image at this location.

Grid calibration

Grid calibration uses an array of uniformly-spaced dots to characterize the lens distortion over the entire field of view. Tangential distortion may optionally be characterized, as well. Grid calibration fits a 2-D polynomial to the measured

[3] `http://youtu.be/2ycFJJcKbMs` (3:04)

dot locations and therefore reduces the errors associated with the simple point distance calibration. NI Vision Assistant provides convenient tools to evaluate the performance of the correction and to select an appropriate balance between performance and computational effort.

► Study the video *Grid Calibration*[4] to learn how to print a dot grid target and then calibrate the camera according to the dot grid.

► Print the dot grid on a laser printer. Measure the distances between the alignment marks to confirm that the dot grid has been printed accurately. Measure in both the horizontal and vertical directions.

► Smooth the paper to be as flat as possible on the camera copy stand base.

$\square_{10}$ Follow along with the steps in the video to set up a grid calibration for your camera. Record the percent distortion and mean error for the five available distortion models.

> **NOTE**: *The camera calibration step must be repeated* any *time that you change the camera configuration. Also be aware that the calibration is only valid for objects that are in the same plane as the dot grid, i.e., calibrating on the dot grid and later refocusing on a much thicker object will yield incorrect results.*

$\square_{11}$ Try some measurements of your ruler. Include ruler positions at the center of the image and at the edges where the distortion corrections are more noticeable. How do these measurements compare to the mean error value reported by the grid calibration procedure?

► Intentionally introduce tangential distortion by tilting the camera relative to the object plane; expect to see *perspective distortion* as imaginary lines intersecting the dot grid now converge rather than remaining parallel.

$\square_{12}$ Re-calibrate your camera and select the option to correct tangential distortion. Screen capture both the uncorrected and corrected images for comparison.

$\square_{13}$ Again, try some measurements of your ruler. Discuss the ability of grid calibration to deal with perspective distortion.

[4] `http://youtu.be/eWqq65ZZ0NQ` (6:47)

5 Stereo Vision

5.1 Synopsis

A *stereo vision* system views the same scene with two cameras at different vantage points to calculate depth information by evaluating the *disparity* between the two views for common edge and texture features. NI Vision provides two example desktop VIs to demonstrate its stereo vision capability, and one of the demos works on live video streams from two webcams.

In this project you will learn how to set up your stereo vision apparatus, calibrate the cameras, and view the depth information as a pseudocolor image.

5.2 Objectives

1. Calibrate two webcams for stereo vision
2. View the rectified webcam images
3. View the depth image
4. Compare the depth as measured by the stereo vision system to the true depth

5.3 Deliverables

1. Screen captures of the desktop VI front panels
2. Lab report or notebook formatted according to instructor's requirements

5.4 Required Resources

1. NI LabVIEW with NI Vision

2. USB webcam and tabletop camera copy stand
3. Dual-camera mounting bracket
4. Dot grid
5. Foam-core board
6. Plastic bricks

5.5 Preparation

NI Vision includes a number of LabVIEW VIs to work with stereo webcams as well as two example VIs to demonstrate these VIs working together to process two webcam video streams to produce a depth image and associated cursor to measure the depth at each point in the scene.

► Review the theoretical concepts and terminology for stereo vision: Open the NI Vision installation folder, e.g., `C:\Program Files (x86)\National Instruments\Vision`, open the `Documentation` subfolder, double-click the `NIVisionConcepts.chm` help file, and expand the hierarchy to select `Machine Vision | Stereo Vision | Stereo Vision Concepts`.

► Study the help file for the example VI Calibrate Stereo Vision System: In the same folder, double-click the `IMAQVision.chm` help file, expand the hierarchy to select `Machine Vision VIs | Stereo Vision VIs`, and then select both `Stereo Vision Example` (this a renamed version of the Calibrate Stereo Vision System VI) and `Stereo Vision FAQs`.

► Locate the two example VIs: Open the NI LabVIEW installation folder, e.g., `C:\Program Files (x86)\National Instruments\LabVIEW 2014`, and then open the subfolder `examples\Vision\Stereo Vision`. You should see two subfolders `Calibrate Stereo Vision System` and `Compute Depth Image`. Open each of these folders and confirm that you see a VI that you can run.

> **NOTE**: *A modified version of the Calibrate Stereo Vision System VI is available in the NI myRIO Vision Essentials Guide*[1] *download file. This modified version works with the table-top camera copy stand so that the telescoping camera column may placed in back so as not to obstruct access to the baseboard.*

[1] `http://www.ni.com/myrio/vision-guide`

► Locate the images used by the Compute Depth Image VI: Back up one level to the `examples\Vision` subfolder, identify the VI IMAQ Vision Example folder, and open and run the VI to see the folder where NI Vision has installed all of its sample images. Navigate to this folder, open the `Stereo Vision` subfolder, and take a look at the images in each of the four subfolders for the two stereo cameras. Note that the calibration folders contain five different views of a dot grid calibration target. Also, can you observe the difference (disparity) between the left and right camera views of the wooden blocks images?

5.6 Compute a Depth Image (Static Images)

► Run the Compute Depth Image VI that you located earlier. See the video tutorial *"Compute Depth Image" Example VI*[2] to get a sense of what you should expect to see.

□$_1$ What is the distance from the camera to poster board on which the blocks rest?

□$_2$ What is the thickness of a block?

□$_3$ What is the tallest pile of blocks that you could find in all available stereo image pairs?

5.7 Compute a Depth Image (Live Images)

► Print the dot grid on a laser printer: Open the NI Vision installation folder, e.g., `C:\Program Files (x86)\National Instruments\Vision`, open the `Documentation` subfolder, and then open the `CalibrationGrid.pdf` file. Choose the "Actual size" option when printing. Measure the distances between the alignment marks in both the horizontal and vertical directions to confirm that the dot grid has been printed accurately.

► Trim the grid to make a 20×15 array of dots, and then affix to a piece of foam-core board or similar material. The board must be as flat as possible to ensure accurate camera calibration.

[2] `http://youtu.be/TEihyEGrsqQ` (3:35)

► Set up your stereo vision apparatus to match that shown in Figure 5.1. The camera-to-camera baseline distance B is 6 cm and the camera-to-baseboard distance D is 24 cm.

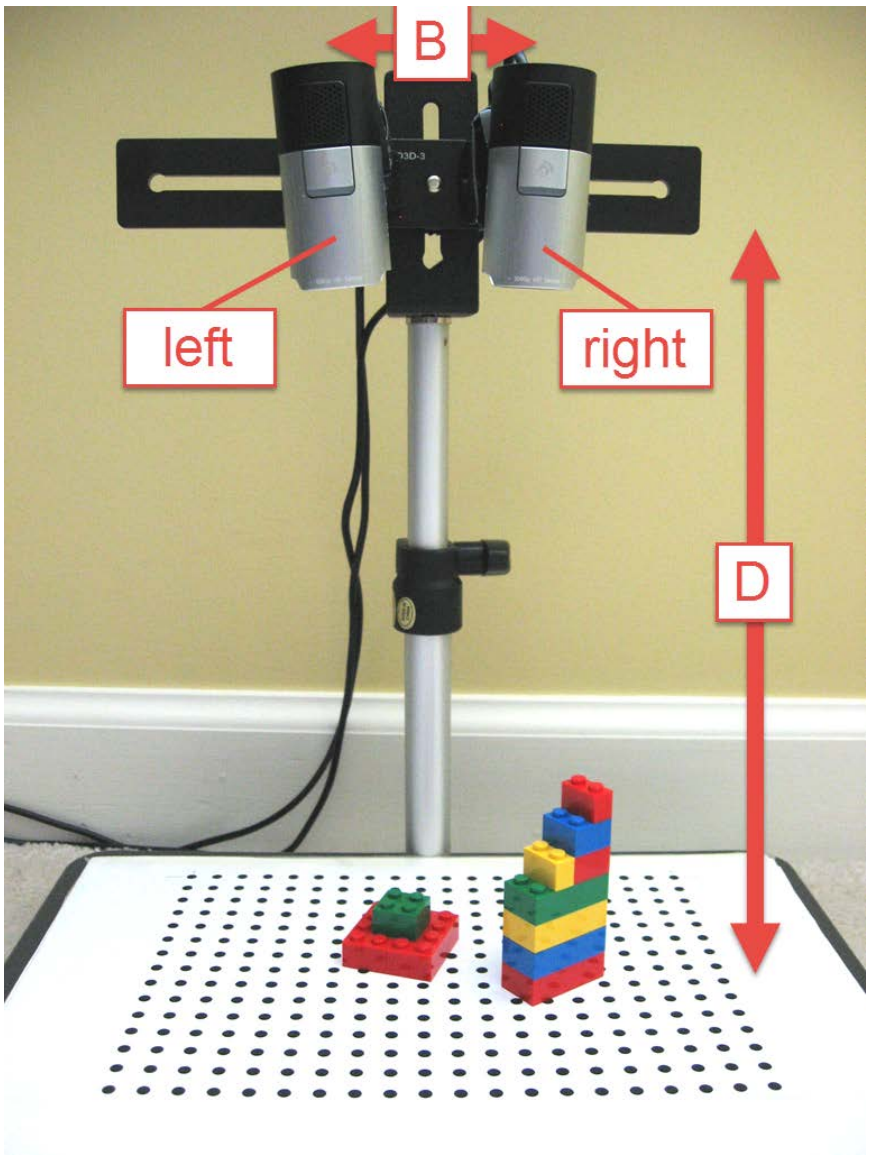

Figure 5.1: Stereo camera setup: $D =$ camera-to-baseboard distance and $B =$ camera baseline.

► Open the Calibrate Stereo Vision System (mod) VI available in the *NI myRIO Vision Essentials Guide*[3] distribution file. See the video tutorial *Configure Stereo Cameras*[4] to learn how to configure the two Vision Acquisition Express VIs for your two webcams. Figure 5.1 on the preceding page shows what is meant by the "Left" and "Right" cameras when mounted to the tabletop copy stand. Use manual settings as much as possible, especially for focus, and use 640×480 resolution.

> **IMPORTANT**: *Think of the two cameras as your two eyes looking out towards a scene. The "left" camera is in the same position as your left eye. The Calibrate Stereo Vision System (mod) VI designates the "Left" Vision Acquisition Express VI to match the "left" camera shown in Figure 5.1 on the facing page.*

► Study the video tutorial *Calibrate Stereo Cameras*[5] to learn how to calibrate the stereo cameras with your dot grid target. Save the calibration file so that you need not repeat this step.

► Use plastic blocks of known thickness to build some shapes similar to those shown in Figure 5.1 on the preceding page. A "step wedge" makes a good subject to verify that the measured depth matches the true depth.

► Study the video tutorial *Measure Depth*[6] to learn how to create a depth image and how to take measurements.

□$_4$ Duplicate the step wedge results from the tutorial video. Report the measured thickness of each step compared to the actual thickness.

□$_5$ Investigate the degree to which translation and rotate of the step wedge impacts measurement accuracy.

□$_6$ Create additional objects and take screenshots of the rectified images and the corresponding depth image.

---

[3] `http://www.ni.com/myrio/vision-guide`
[4] `http://youtu.be/iGqlBrg3DjM` (2:47)
[5] `http://youtu.be/bufXESBlzTY` (2:49)
[6] `http://youtu.be/UXw31LOSiHk` (3:16)

6 Coin Caliper I

Anytime you begin to learn a new body of material it is easy to become overwhelmed by all of the details. In this first lab project you will concentrate only *on learning how to "drive" the software tools to make a simple application to measure the diameter of a circular coin. Later projects will introduce the machine vision concepts that can be implemented by these tools.*

Objective: The *Coin Caliper* accepts an image of a single circular coin and reports the coin's measured diameter in real-world units (millimeters).

Technical Approach: The NI Vision *Max Clamp* step automatically detects the boundary of an object and measures its maximum height or width; for a circular object such as a coin *Max Clamp* measures the diameter. With a properly calibrated camera this diameter can be reported in millimeters.

6.1 Required Resources

1. NI LabVIEW and NI Vision Assistant
2. NI myRIO with power supply and USB cable
3. USB webcam and tabletop camera copy stand
4. Black velvet paper background
5. Ruler with millimeter and centimeter marks
6. Several coins of various sizes
7. Caliper

Deliverables:

1. Lab report or notebook formatted according to instructor's requirements
2. .zip archive file containing all project files (acquired images and LabVIEW project)

6.2 Procedure

Configure the image acquisition system

$\square_1$ Study the video *Set Camera Defaults in NI-MAX*[1] to learn how to use NI-MAX to select the camera resolution, frame rate, and output image type as either color or grayscale, and then save these settings as the default for other applications. The default settings may be defined independently for the same webcam connected to the desktop and to NI myRIO. Also learn how to determine the frame rate for the displayed and acquired images for a given video mode and output image type.

$\square_2$ Connect your webcam to the desktop and then record and tabulate the frames per second (fps), both displayed and acquired, as reported by NI-MAX for each of available video modes of your webcam as well as the two types of output (color and grayscale), if available. Connect the webcam to NI myRIO and repeat; the video modes are likely to be different on the desktop than those on NI myRIO. At what resolution do you begin to see a difference between RGB and grayscale frame rate? Does the video mode with the highest stated frame rate for a given resolution in fact have the highest *measured* frame rate? Also investigate the impact on frame rate of manual versus automatic settings, especially exposure.

> **TIP**: *Keep this table handy for future machine vision projects so that you can quickly choose a video mode to balance the competing requirements of throughput (frame rate) and spatial resolution.*

Acquire representative images

$\square_3$ Gather at least five coins of different denominations, and at least three additional coins of a single denomination.

[1] `http://youtu.be/oTWHhMENsCg` (3:51)

$\square_4$ Determine as accurately as possible the diameters of the coins you have available. Precision calipers would be appropriate, if available, otherwise search the Internet for the specifications of your coins. Tabulate the diameters of each of your coin denominations.

$\square_5$ Follow along with the video *Step 1: Obtain Representative Images*[2] to learn how to acquire representative coin images in the stand-alone version of NI Vision Assistant; select Windows "Start" and then select "All Programs | National Instruments | Vision | Vision Assistant." Adjust your webcam setup to obtain images similar to those presented in the video, and choose a video mode with reasonably high resolution. Remember to set the focus to manual.

> **TIP**: *The NI Vision Assistant "Acquire Images" tab defaults to "This Computer" as the target device and displays a list of all desktop-connected cameras from which you can choose. You can also access cameras connected to NI myRIO within Vision Assistant: Choose "Select Network Target" instead of "This Computer" and then enter the IP address as either* `172.22.11.2` *when myRIO is connected by its USB cable or the IP address associated with myRIO by the wireless network.*

$\square_6$ Collect images of at least five different coins and at least three additional images of the same coin denomination (to evaluate measurement repeatability). Also collect an image of a metric ruler as a calibration target. Record the video mode and camera settings that you adjusted. Capture a screenshot of the Vision Assistant image browser that shows all of your images at once.

$\square_7$ Remember to save all of your images to disk.

Calibrate the camera

$\square_8$ Follow along with the video *Step 2: Calibrate the Camera*[3] to learn how to perform "Point Distance Calibration," the simplest available calibration that ignores lens distortion and other non ideal effects. Calibrate the camera in the center of the image where the coins will be placed.

[2] `http://youtu.be/DMH5Z-J0mcA` (4:21)
[3] `http://youtu.be/2ycFJJcKbMs` (3:04)

$\square_9$ Use the distance measurement tool to measure the length between two ruler marks in the center of the image and report the percent error. Repeat for two other lengths near the left and right edges of the image. Comment on the measurement accuracy in the image center compared to the edges.

Develop the processing script

$\square_{10}$ Follow along with the video *Step 3: Develop Vision Script*[4] to learn how to create a processing script based on the *Max Clamp* virtual calipers to measure the diameter of a single coin presented to the region of interest (ROI) in the center of the image.

$\square_{11}$ For each coin image record and tabulate the measured coin diameter reported by *Max Clamp*, the actual coin diameter, and the percent error of the measurement.

Develop the base LabVIEW myRIO project

$\square_{12}$ Follow along with the video *Step 4: Create LabVIEW Project*[5] to learn how to create a new LabVIEW project for the NI myRIO target.

> **NOTE**: *You can also use the desktop as your target if you do not have access to NI myRIO. Simply create a new LabVIEW VI or project as usual and then continue with the same development flow.*

$\square_{13}$ Connect the webcam to the NI myRIO USB port.

$\square_{14}$ Follow along with the video *Step 5: Configure "Vision Acquisition" Express VI*[6] to learn how to configure the Vision Acquisition Express VI for continuous in-line processing using the same video mode and camera settings used previously to acquire the representative images. Enable the frame rate indicator output.

$\square_{15}$ Screen capture the front panel diagram of a single coin in the display. How well does the image match your previously acquired images? What frame rate is reported?

---

[4] `http://youtu.be/AO5Kh-VN5sA` (2:59)
[5] `http://youtu.be/zZ1oLiItM3g` (1:31)
[6] `http://youtu.be/DmUHugQ6qJM` (3:34)

$\square_{16}$ Follow along with the video *Step 6: Configure "Vision Assistant" Express VI*[7] to learn how to configure the Vision Assistant Express VI with the processing script that you developed earlier, check the script performance with the "Performance Meter," install the Express VI inside the while-loop structure, and attach controls and indicators. Run the VI and debug the code until the application works properly.

$\square_{17}$ Screen capture the front panel for several coins. Compare the measured diameters to the known diameters. Report the typical frame rate, too.

$\square_{18}$ Follow along with the video *Step 7: Add I/O Devices*[8] to learn how to add the NI myRIO on-board LEDs to indicate whether the coin diameter is larger or smaller than a threshold value. This very simple example illustrates how to add I/O devices to the application.

Validate the application

$\square_{19}$ Test your completed "Coin Caliper" application. How robust is the application? For example, how much variation do you see in the diameter measurement with the coin "just sitting there" in the display? How do changes in illumination affect the measurement?

Deploy the VI as a stand-alone application

$\square_{20}$ Study the video *Deploy a Stand-Alone Application*[9] to learn how to deploy the "Coin Caliper" as a stand-alone application on NI myRIO. After completing this process the "Coin Caliper" will run automatically when you power-on the NI myRIO.

Add an external information display

$\square_{21}$ Extend the dual-LED display technique described in the most-recent video to use all four LEDs as a bargraph indicator for the measured coin diameter. Review your coin collection and determine the minimum coin diameter $D_{\min}$ and the maximum coin diameter $D_{\max}$. Disable LEDs for a measured diameter less than $D_{\min}$, enable all LEDs for a measured diameter greater than or equal to

[7] `http://youtu.be/_il9DoPwkpM` (4:37)
[8] `http://youtu.be/0RAIwdW5byk` (3:06)
[9] `http://youtu.be/JXoJECRS-eo` (8:29)

$D_{\max}$, and illuminate a proportional number of LEDs for a measured diameter between $D_{\min}$ and $D_{\max}$.

7 Coin Caliper II

Learn how to adapt the "Machine Vision App" template to implement a more sophisticated version of the "Coin Caliper" application you created in the previous project. Follow along with the step-by-step tutorials to gain experience with the various template editing techniques.

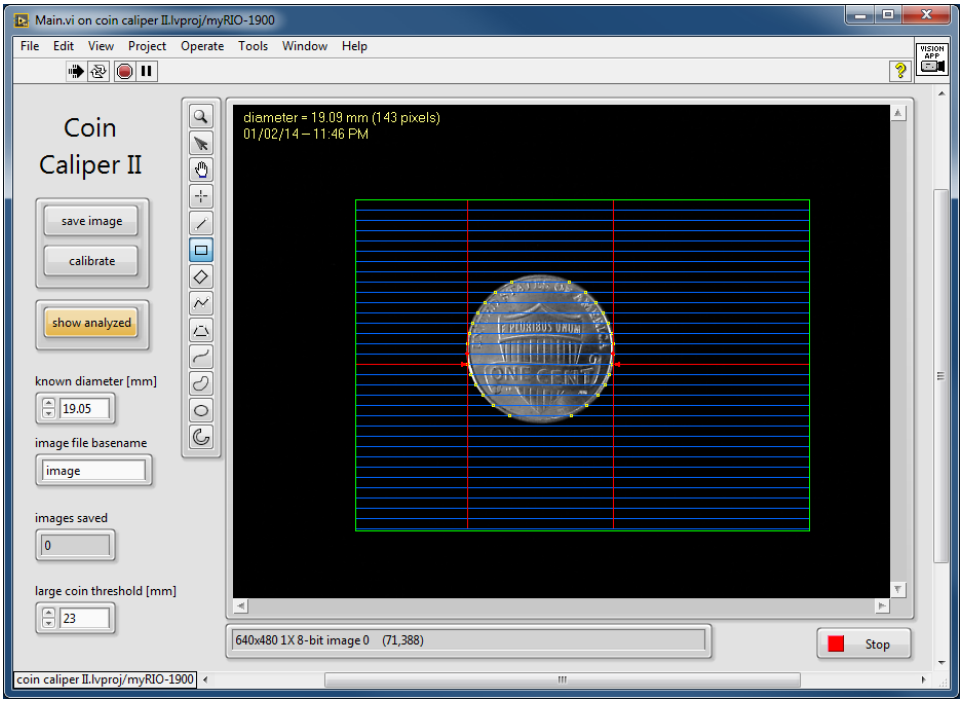

Figure 7.1: Front panel of the *Coin Caliper II* VI.

Functional Requirements: The *Coin Caliper II* will:

1. Accept an image of a single circular coin contained in a user-defined region of interest (ROI),
2. Measure the coin's diameter in both pixels and real-world units (millimeters),
3. Report the coin's diameter in both pixels and in real-world units as well as the current time and date as a nondestructive image overlay,
4. Display the analyzed image when show analyzed is pressed, otherwise display the webcam image,
5. Recalculate the real-world units scaling factor based on a coin of known diameter when calibrate is clicked,
6. Save the currently-displayed image to a USB flash drive when save image is clicked or when the NI myRIO onboard button is pressed,
7. Define the image file name as a user-defined base name combined with the image sequence number,
8. Display the number of images saved during the current session,
9. Illuminate the NI myRIO onboard LED 0 when a coin is present, and
10. Illuminate the NI myRIO onboard LEDs 1 to 3 when a coin's measured diameter exceeds a user-defined threshold entered on the front panel.

Figure 7.1 on the preceding page shows the front-panel of the *Coin Caliper II* application.

Required Resources:

1. NI LabVIEW and NI Vision Assistant
2. NI myRIO with power supply and USB cable
3. USB webcam and tabletop camera copy stand
4. USB hub (Stratom X-HUB or equivalent)
5. USB flash drive
6. Black velvet paper background
7. Ruler with millimeter and centimeter marks
8. Several coins of various sizes

Deliverables:

1. Lab report or notebook formatted according to instructor's requirements; include front-panel screen shots to demonstrate that your application meets functional specifications
2. .zip archive file containing all project files:
   (a) Representative images
   (b) Vision Assistant script
   (c) LabVIEW project file set

7.1 Technical Approach

NI Vision Assistant Script

Clamp Rake serves as the heart of the Vision Assistant script for this project. It requires a grayscale image (see *Extract Luminance Plane*[1] to learn how to extract the luminance plane from a color image) and seeks two edges either horizontally or vertically that define the maximum extent of an object. *Clamp Rake* effectively measures the diameter of circular object such as a coin. Refer to *Measure Width with "Clamp (Rake)"*[2] to learn how to configure this step. Select the "ROI Descriptor" control and the two "Distance" indicators when you import the script into the LabVIEW Vision Assistant Express VI.

A simple 2-point calibration with the *Image Calibration* step suffices to convert pixel measurements from *Clamp Rake* to real-world units in millimeters. Refer to *Two-Point Calibration from a Ruler*[3] for details. Select the "Grid Descriptor" control for this step when you import the script.

LabVIEW Techniques

The following LabVIEW techniques are required for this project:

- *Get ROI from Image Display*[4] – Get the ROI (region of interest) from the main image display with a "Property Node."
- *Add Custom Data Lane*[5] – Add a custom lane: create a control from the Vision Assistant Express VI indicator output, convert to type def, save

[1] `http://youtu.be/1mmzsd46_Gs` (1:02)
[2] `http://youtu.be/fPWQCnB7B4s` (2:18)
[3] `http://youtu.be/kx45UGkbbbo` (2:28)
[4] `http://youtu.be/sKeO2iM8UWE` (2:29)
[5] `http://youtu.be/GkqjZCj3xck` (2:58)

type def as a control, add the control to the data highway, merge the Vision Assistant Express VI output onto the data highway, and then retrieve the value in another state.

- *Create Formatted Text String*[6] – Create a formatted text string with the "Format Into String" VI. The syntax is similar to the "printf" function in C.
- *Get Date and Time Strings*[7] – Get the system date and time as strings using the "Get Date/Time in Seconds" and "Get Date/Time String" VIs.
- *Overlay Text*[8] – Create a nondestructive text overlay with the "IMAQ Overlay Text" VI.
- *myRIO Onboard Button*[9] – Place the NI myRIO onboard button Express VI.

7.2 Procedure

Follow along with this step-by-step procedure to convert the functional requirements listed above into a working application.

1. *Step 1: Rename and Open Project*[10] – Extract the "Machine Vision App" (MVA) template from the .zip file downloaded from the *NI myRIO Vision Essentials Guide*[11] site, rename the folder, rename the .lvproj file, open the project, rename the while-loop structure, and rename the app.
2. *Step 2: Interpret Specifications*[12] – Interpret the functional specifications to develop the high-level design of the application:
   (a) Interpret the specifications to look for tasks, action buttons, mode buttons, and default behavior,
   (b) Design the state sequence for the default task,
   (c) Make note of new data highway lanes, and
   (d) Create a task diagram (e.g., Figure 7.2 on page 51) and diagram or tabulate the state sequence for each task as well as the detailed activities within each state:

[6] `http://youtu.be/VqH0SSK1fHY` (1:44)
[7] `http://youtu.be/U4aNrolgDMA` (1:02)
[8] `http://youtu.be/SS72-mKX6fQ` (2:15)
[9] `http://youtu.be/Sa4SsFWgVew` (0:57)
[10] `http://youtu.be/YzbmvF5ZLUA` (1:22)
[11] `http://www.ni.com/myrio/vision-guide`
[12] `http://youtu.be/sbP45Merxxs` (4:45)

i. ***Calibrate*** (Figure 7.3 on page 52),
ii. ***Save Image*** (Figure 7.4 on page 53, and
iii. Default task (Figure 7.5 on page 54

3. *Step 3: Develop Vision Script*[13] – Coin Caliper II demo Step 3/15: Develop the vision script in the stand-alone version of Vision Assistant to measure and report the coin diameter in pixels and in real-world units.
4. *Step 4: Import Vision Script*[14] – Coin Caliper II demo Step 4/15: Import the vision script into the LabVIEW Vision Assistant Express VI; select controls and indicators.
5. *Step 5: Connect Script*[15] – Coin Caliper II demo Step 5/15: Connect the Vision Assistant Express VI controls and indicators to the data highway.
6. *Step 6: Obtain ROI*[16] – Coin Caliper II demo Step 6/15: Use a "Property Node" to obtain the user-selected ROI from the main image display.
7. *Step 7: Create Task Buttons*[17] – Create an action button for the ***calibrate*** state.
8. *Step 8: Create Increment State*[18] – Create the ***increment file number*** state and add "file number" to the data highway.
9. *Step 9: Create Calibrate State*[19] – Create the ***calibrate*** state, add "known diameter" to the data highway, and calculate the scale factor as known diameter in millimeters divided by measured diameter in pixels.
10. *Step 10: Edit Overlay State*[20] – Edit the ***(qsm) overlay*** state to add the coin diameter to the nondestructive overlay.
11. *Step 11: Edit Save Image State*[21] – Edit the ***save image*** state to use "file number" from the data highway.
12. *Step 12: Edit myRIO Ports State*[22] – Edit the ***(myRIO) update ports*** state to operate the myRIO LED indicators.

[13] `http://youtu.be/LR33-W4Thww` (5:12)
[14] `http://youtu.be/sIj-EPxV8Tg` (2:24)
[15] `http://youtu.be/TWYBgLmCmnc` (3:53)
[16] `http://youtu.be/LzvU9eOcXO8` (1:32)
[17] `http://youtu.be/8WKzPgNUbQ4` (1:07)
[18] `http://youtu.be/5yp9NaZA7SI` (2:45)
[19] `http://youtu.be/pc2DZfITPrM` (2:43)
[20] `http://youtu.be/DgJ-ere6_5k` (2:19)
[21] `http://youtu.be/JwuVK83GLJo` (4:40)
[22] `http://youtu.be/7LFFkM7CZBg` (2:42)

13. *Step 13: Edit Scheduler State*[23] – Edit the ***(qsm) schedule*** state to create the necessary tasks.
14. *Step 14: Set Up Camera*[24] – Coin Caliper II demo Step 14/15: Set up the camera mode (resolution) and attributes.
15. *Step 15: Run and Debug*[25] – Coin Caliper II demo Step 15/15: Run and debug the application until it works properly.

Congratulations! You have now implemented your first machine vision application using the "Machine Vision App" template project, and you are now prepared to move on to more advanced projects.

[23] `http://youtu.be/jLb0lOpKnx0` (2:17)
[24] `http://youtu.be/jEXkhzawvNE` (2:54)
[25] `http://youtu.be/3gbpGOIED5U` (8:19)

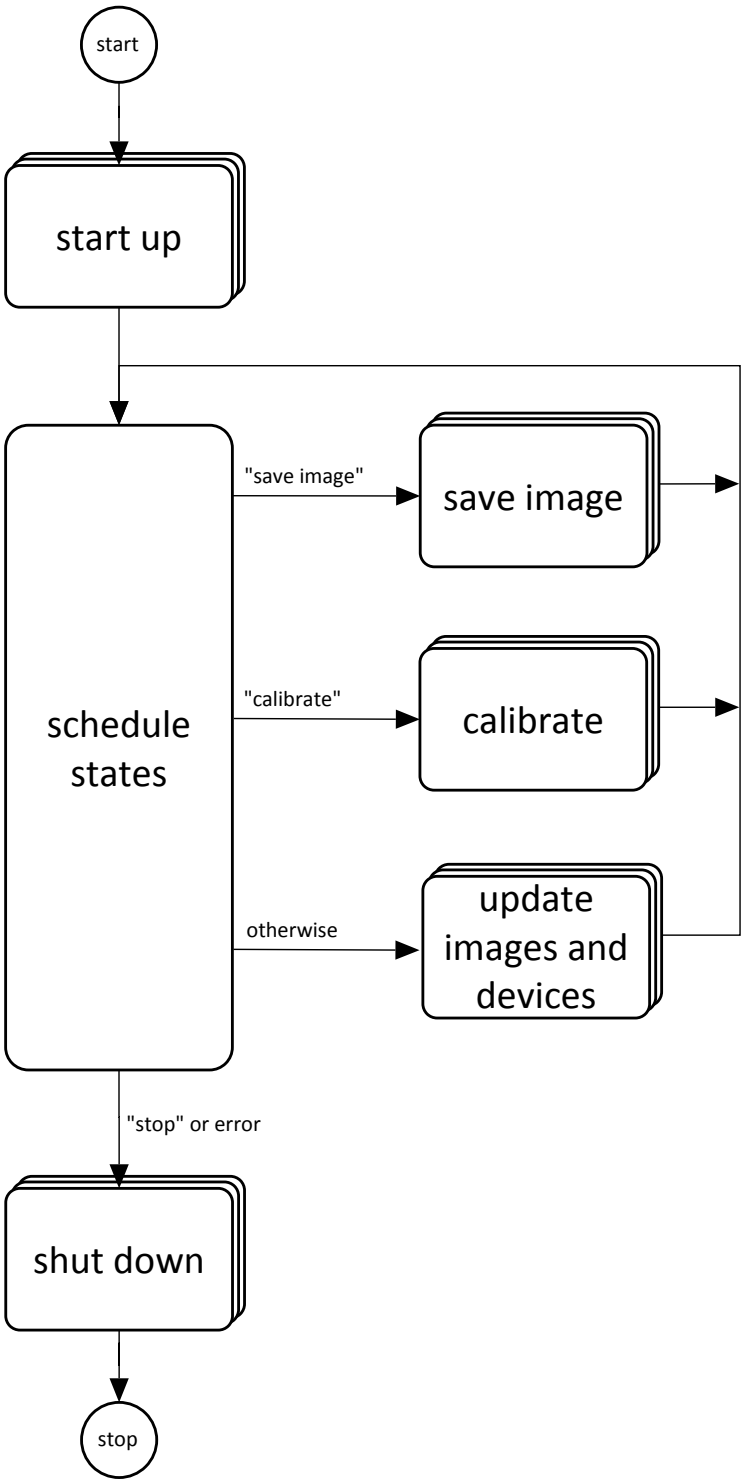

Figure 7.2: Task diagram for the *Coin Caliper II* application.

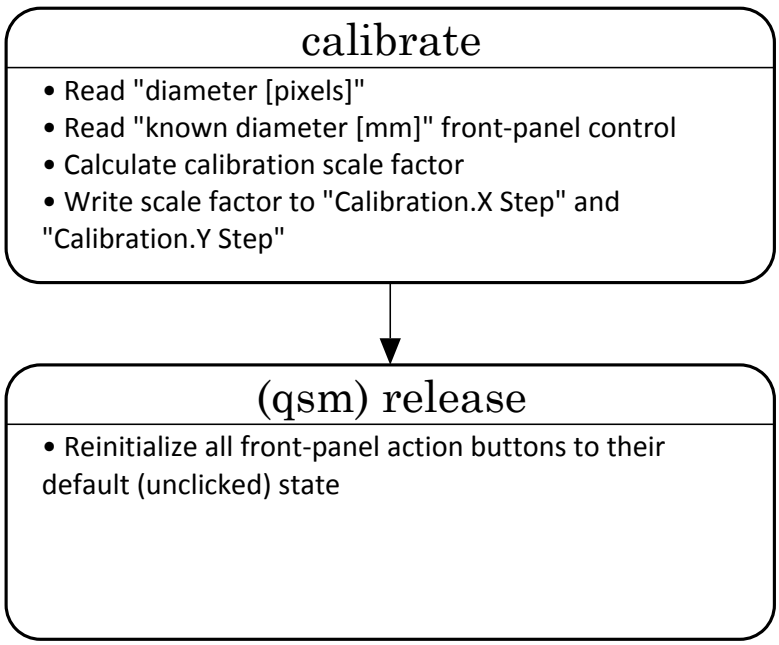

Figure 7.3: ***Calibrate*** task for the *Coin Caliper II* application.

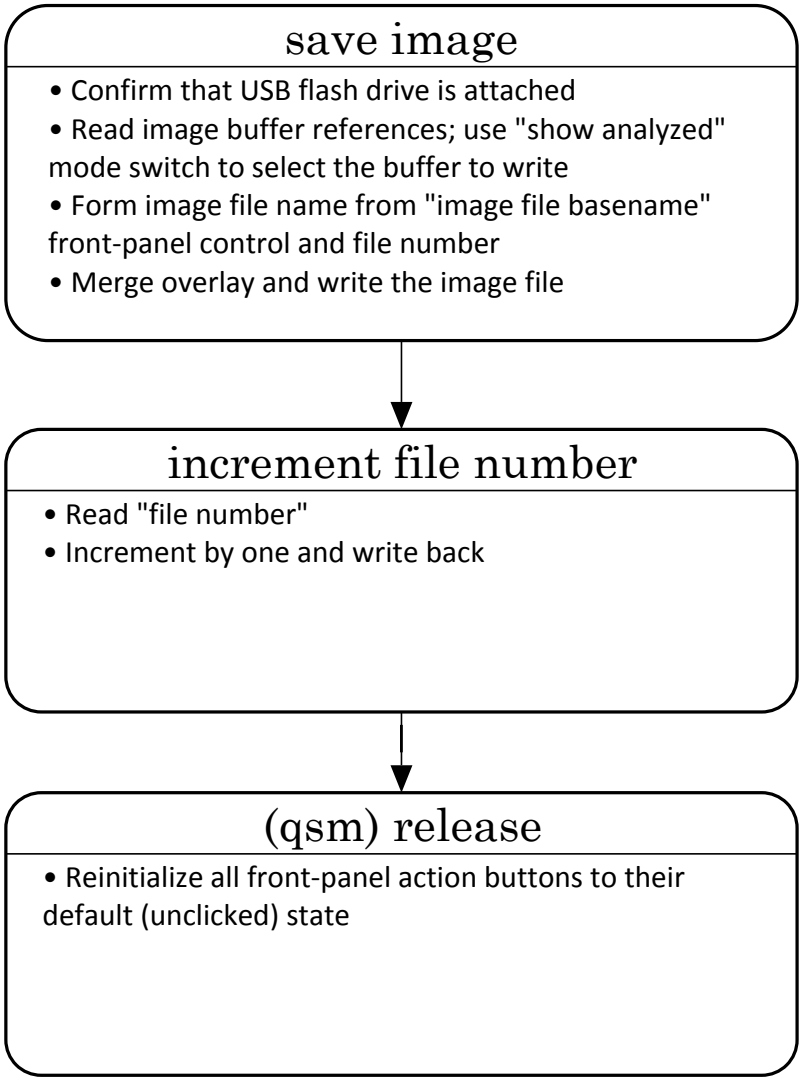

Figure 7.4: ***Save Image*** task for the *Coin Caliper II* application.

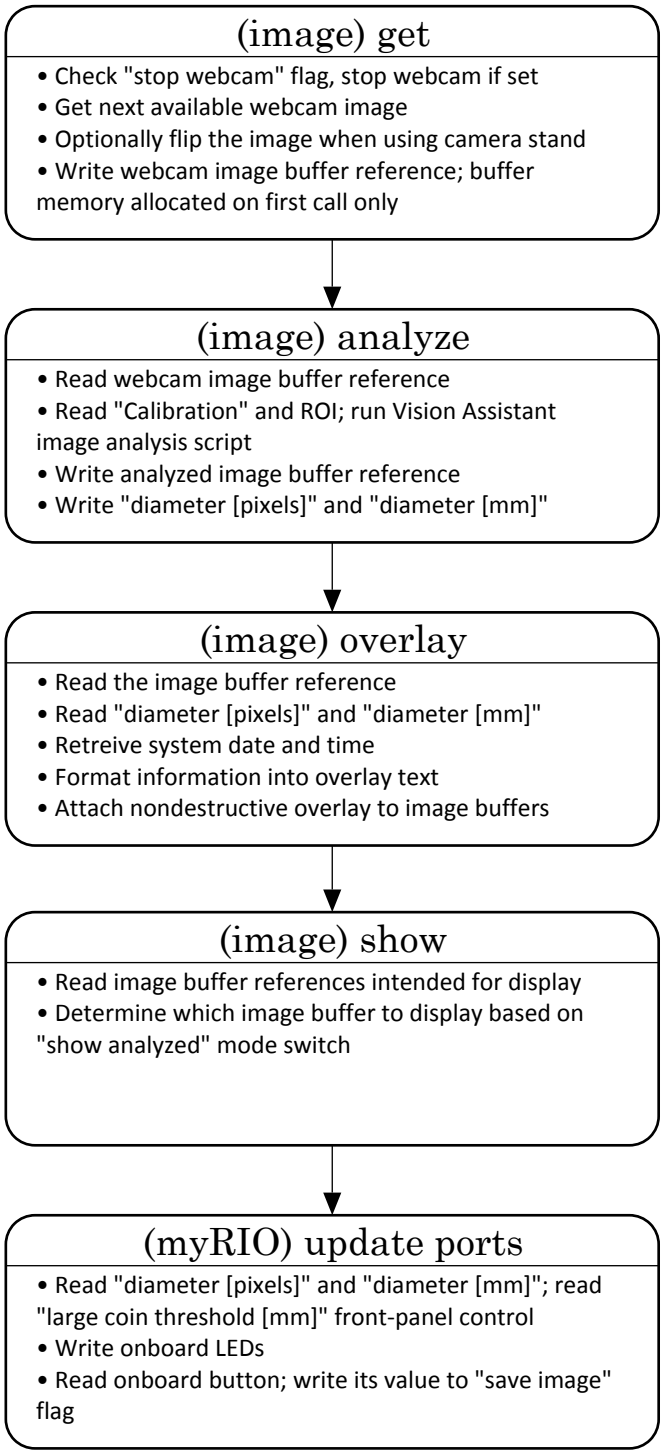

Figure 7.5: Default task for the *Coin Caliper II* application.

Part III

Machine Vision Application Projects

8 Coin Counter

Do you have piles of loose change accumulating here and there? Does counting all of those pennies seem like too much bother? Save a trip to the bank and create your own machine vision application to "look" at a collection of coins and instantly show you the total value!

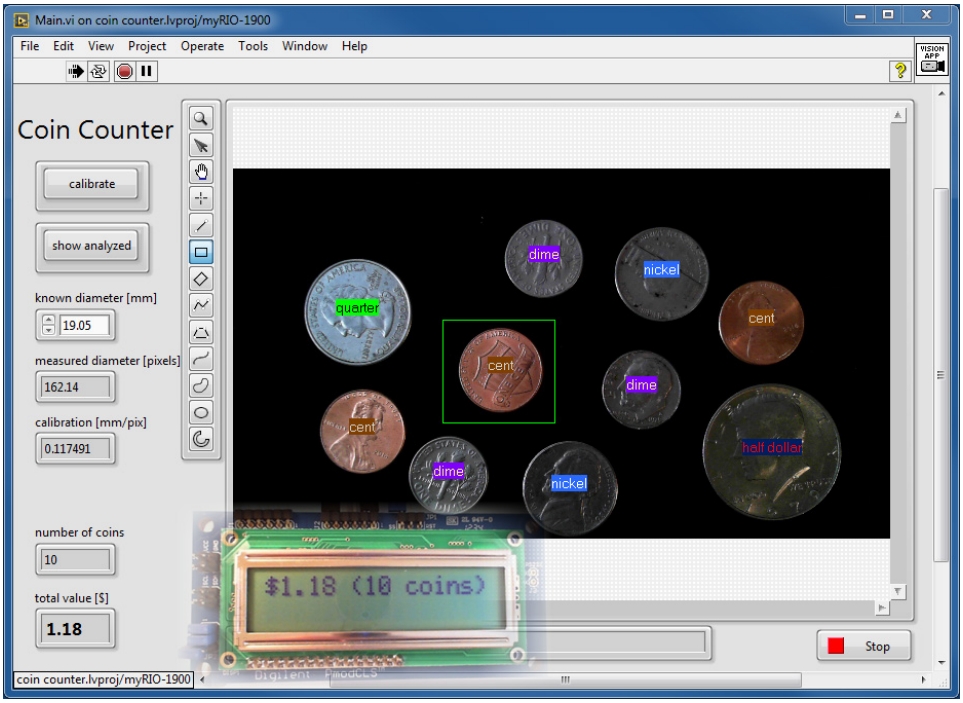

Figure 8.1: Front panel of the *Coin Counter* application.

Functional Requirements: The *Coin Counter* will:

1. Accept an image of an arbitrary number of coins of at least four classes (denominations) of coins,
2. Accommodate up to 15 coins in the field of view (FOV),
3. Report each coin's denomination as a text label superimposed over the coin,
4. Display the number of coins and total value both on the front panel as well as on an external LCD display,
5. Recalculate the real-world units scaling factors based on a coin of known diameter selected by a user-defined ROI when calibrate is clicked,
6. Display the analyzed image when show analyzed is pressed, otherwise display the webcam image,
7. Display the measured diameter of the coin used for calibration, and
8. Display the calibration scale factor.

Figure 8.1 on the previous page shows the front-panel of the *Coin Counter* application. Note the use of distinct foreground and background colors for the overlay text labels, as well as the user-defined ROI around a coin of known diameter. Figure 8.2 on the facing page shows the image produced by the Vision Assistant script.

Required Resources:

1. NI LabVIEW and NI Vision Assistant
2. NI myRIO with power supply and USB cable
3. USB webcam and tabletop camera copy stand
4. Black velvet paper background
5. LCD display (Digilent PmodCLS)
6. Coins – ten samples each of at least four coin classes (denominations)

Deliverables:

1. Lab report or notebook formatted according to instructor's requirements; include front-panel screen shots to demonstrate that your application meets functional specifications

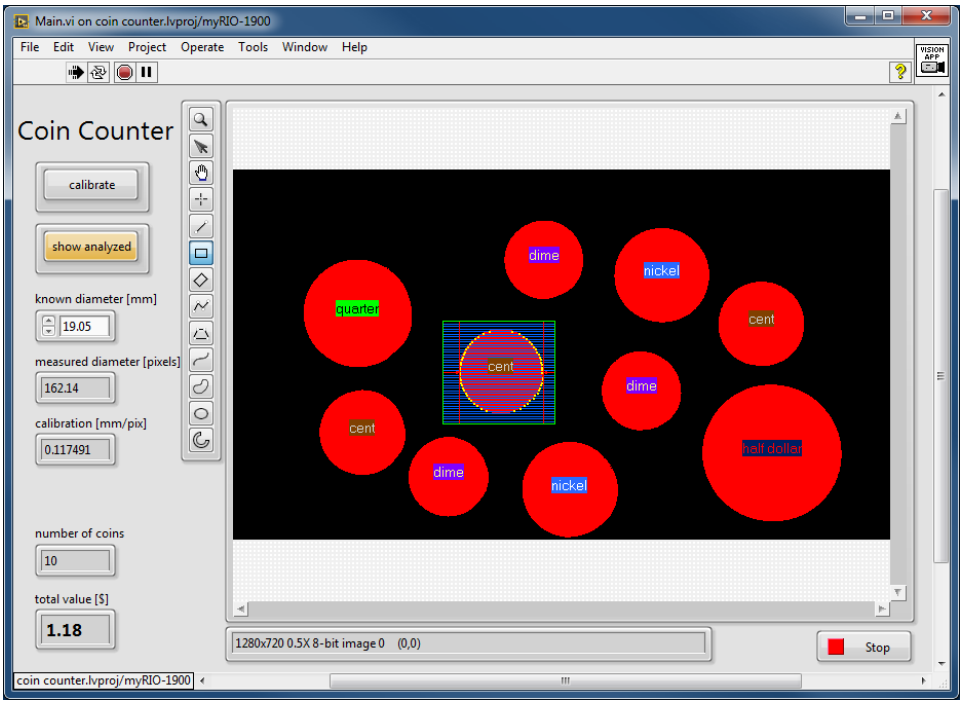

Figure 8.2: Front-panel for the *Coin Counter* application showing the output of the Vision Assistant script.

2. .zip archive file containing all project files:

   (a) Representative images
   (b) Vision Assistant script
   (c) LabVIEW project file set

8.1 Technical Approach

NI Vision Assistant Script

NI Vision Assistant provides the *Circle Detector* step as a powerful yet simple-to-use method to detect circular objects in a binary (two-level) image and report the center coordinates and radius of each detected circle in the image. Comparing the measured circle diameter (two times the measured radius) to known threshold values that separate the coin classes identifies the coin and permits the total value of all coins in the FOV to be calculated. The detector requires a

clean image free of extraneous features, and the morphological operations "remove small objects" and "fill holes" available under the *Advanced Morphology* step work well for this purpose. The *Threshold* step offers a wide variety of auto-thresholding operations to separate the foreground coin objects from the background, and the auto-thresholding feature allows the thresholding step to work reliably under changing lighting conditions.

You will find it convenient to set up your coin class diameter thresholds as constants in the ***(qsm) initialize*** state based on real-world units such as millimeters instead of pixels. Calibrate the image using the general technique that you developed for the *Coin Caliper II* application (Chapter 7 on page 45), e.g., image a coin of known diameter to recalculate the calibration scale factor with an action button. As of this writing the *Circle Detector* step only provides measurements in pixels, therefore you may omit the *Image Calibration* step and simply use *Clamp Rake* to measure the diameter in pixels for a single selected coin; for this reason remember to select the "ROI Descriptor" control and the "Distance" indicator when you import the script into LabVIEW. Once you have filled the FOV with coins you need to be able to draw an ROI around a single coin instead of physically removing all but one coin.

Study these short video tutorials to learn how to use each of the recommended NI Vision Assistant steps for this project:

- *Circle Detector* – *Binary Circle Detector*[1] – learn how to configure the range of acceptable radii and how to export measurements to an Excel file for further analysis,
- *Advanced Morphology* – learn how configure the various operations such as "remove small objects" (*Remove Small Objects*[2]) and "fill holes" (*Fill Holes*[3])
- *Threshold* *Threshold Grayscale to Binary*[4]– learn how to manually threshold an image or select from a variety of auto-thresholding routines.
- *Measure Width with "Clamp (Rake)"*[5] – learn how to measure the diameter of a single coin within the ROI.

[1] `http://youtu.be/9fo76IeeUMM` (1:47)
[2] `http://youtu.be/Xfo9wK_4B7Q` (1:09)
[3] `http://youtu.be/GjwostpEeTo` (1:35)
[4] `http://youtu.be/7fJBS6_neQY` (3:08)
[5] `http://youtu.be/fPWQCnB7B4s` (2:18)

Digilent PmodCLS LCD Display

Chapters 26 to 28 of the *NI myRIO Project Essentials Guide*[6] provide a detailed explanation of the Digilent PmodCLS LCD display using its UART, SPI, or I^{2}C-bus interface as well as LabVIEW demonstration code available at *NI myRIO Project Essentials Guide .zip Files*[7] that you can drop into this project. Study *LCD Character Display Interfacing Theory*[8] to learn how to write escape sequences to configure and control the LCD display and then study one of these block diagram "walk-through" videos depending on the serial bus that you wish to use: *LCD (UART) Demo Walk-Through*[9], *LCD (SPI) Demo Walk-Through*[10], or *LCD (I2C) Demo Walk-Through*[11].

> **TIP**: *Refer to the NI myRIO connector diagrams in Appendix B on page 137.*

LabVIEW Techniques

The following LabVIEW techniques are required for this project:

- *Get ROI from Image Display*[12] – Get the ROI (region of interest) from the main image display with a "Property Node."
- *Add Custom Data Lane*[13] – Create a custom control for the "Circles Data" Vision Assistant script output and insert this into the data highway
- *Create Formatted Text String*[14] – Create a formatted text string with the "Format Into String" VI. The syntax is similar to the "printf" function in C.
- *Overlay Text at X-Y Coordinates*[15] – Programmatically place overlay text at a specific X-Y location.

---

[6] `http://www.ni.com/myrio/project-guide`
[7] `http://www.ni.com/academic/myrio/project-guide-vis.zip`
[8] `http://youtu.be/m0Td7KbhvdI` (10:36)
[9] `http://youtu.be/JsEMMnIWg4k` (3:44)
[10] `http://youtu.be/oOXYryu4Y-c` (4:23)
[11] `http://youtu.be/qbD31AeqOMk` (4:32)
[12] `http://youtu.be/sKeO2iM8UWE` (2:29)
[13] `http://youtu.be/GkqjZCj3xck` (2:58)
[14] `http://youtu.be/VqH0SSK1fHY` (1:44)
[15] `http://youtu.be/O5IIrPP_smA` (1:41)

- *Select Image Palette*[16] – Use a Property Node to set the image display palette to either "Binary" or "Grayscale." Note that the *Circle Detector* step produces a binary output that appears like an all-black screen unless the "Binary" palette is selected.

8.2 Development Tips

Representative Images

- Gather at least ten samples each of four classes of coins, e.g., dime, penny, nickel, and quarter.
- Conduct an Internet search to find the diameter of each coin class. Determine the minimum difference in diameter between all pairs of coin classes in millimeters and record this value as the minimum feature size. Allocate at least 6 pixels to this minimum feature size to account for measurement errors due to perspective distortion and other non ideal effects and then calculate the required spatial resolution.
- Determine the camera distance and camera mode (resolution and frame rate) that achieves your required spatial resolution while maximizing the number of coins that fit within the FOV while also maximizing the frame rate. Strive for a balance of frame rate, FOV size, and accuracy. Later on you may find that you need to make some adjustments to obtain satisfactory results.
- Use black velvet paper as the background to create high contrast with the shiny metal coins, and also to protect the surface of the table-top camera stand. Ambient room light should suffice; keep the lighting as uniform as possible to avoid lighting "hot spots."
- Use a line grid target to align the camera so as to minimize perspective distortion. A properly-aligned camera will minimize measurement errors and avoid the computational cost of image calibration and correction.
- Adjust the camera attributes to maximize the contrast between the foreground objects (coins) and the background (black felt). Use manual settings for all camera attributes, adjusting brightness, contrast, exposure, and so on to make the background as uniformly black as possible; don't worry about overexposing the coins because you will eventually threshold the image to binary and remove the coin features altogether.

[16] `http://youtu.be/M53-QmPsSq4` (2:09)

- Collect images to study *intraclass* variation: place as many samples of a given class as will fit in the FOV without touching or overlap, and then repeat for each class of coin. With these images you will be able to measure the expected spread of measurements due to measurement error.

Vision Assistant Script

- Create an NI Vision Assistant script to convert the image to grayscale, auto-threshold to binary, remove noise, and detect circular objects. As you develop the script remember to confirm correct script operation by occasionally cycling through all of your representative images.
- Select your images that contain multiple samples of a given coin class, export the measured radii for each class to a spreadsheet, sort the radii from low to high, and then look for threshold values to distinguish the classes. For example, if your smallest coin has measured radii in the range 60 to 64 pixels and the next smallest coin has radii from 68 to 72 pixels, then the midpoint value 66 pixels will serve as a threshold to distinguish between these two coin classes.
- You may find that intraclass variation is too high to identify a threshold value, i.e., the measured radii distribution of the two coin classes nearest in size overlaps. If so, begin by adjusting the camera to ensure that you have minimized perspective distortion that artificially enlarges coins at the periphery of the image. If that fails to improve the measurement accuracy you will need to increase the spatial resolution either by moving the camera closer to the object plane or by selecting a higher-resolution camera mode. You may also wish to investigate the *Image Correction* step (requires a real-world calibrated image with grid distortion model, see *Correct a Grid-Calibrated Image*[17] for details); image correction applies the calibration model to every pixel in the image and may reduce the perspective distortion to a manageable level. However, be certain to consult "Tools | Performance Meter" to weigh the additional computational cost.

Coin Classifier

- The *Circle Detector* step produces a cluster array that contains the measured radius and centroid coordinates for each detected circle. The measured radius must be translated into a coin's value, text label, foreground color and background color. Create a custom cluster array control coin info

[17] `http://youtu.be/mJFsqqd37WY` (1:34)

that contains these four elements (see *Create "Coin Info" Cluster Array*[18]), and then create a data highway lane for this control. Create another lane as an array of corresponding known coin diameters. Define these values as constants in the ***(qsm) initialize*** state.

- Create a subVI to produce the coin info record that is closest to the measured coin radius; remember to multiply the radius by two and to account for the calibration scale factor. Calculate the distance between the measured diameter and all of the known coin diameters (subtract the array of known diameters from the measured diameter and then take the absolute value) and then use the Array Max & Min function to find the array index of the minimum distance.

  > **TIP**: *Use the "Quick Drop" shortcut (Ctrl + space bar) to look up and place a LabVIEW element by name. See Appendix C on page 139 for more LabVIEW tips.*

- Create a ***calculate value*** state to sum the individual coin values with a for-loop structure auto-indexed by the "Circles Data" array to step through each of the detected coins. Edit the ***(image) overlay*** state to add a similar for-loop structure to create text labels to be placed at the center of each coin; see *Overlay Text at X-Y Coordinates*[19] and *Adjust Overlay Text Alignment*[20].

8.3 Validation Tips

Evaluate the performance of your completed application, especially these use cases:

- No coins in view,
- One coin moved to various locations – confirm that the coin is always classified properly,
- One sample each for all four classes – confirm that the total value displayed is correct,
- Maximum number of coins in view, various number of samples of each class – again, confirm that the total value is correct, and

[18] `http://youtu.be/Yqv4u5FE8Vk` (5:12)
[19] `http://youtu.be/O5IIrPP_smA` (1:41)
[20] `http://youtu.be/rsI7fco5n_g` (2:47)

- Touching coins, overlapping coins and coins only partly in view – how much overlap can be tolerated before you observe classification errors?

8.4 Optional Extra Features:

You may wish to add one or more of these extra features:

- Blink an on-board LED when no coins are detected.

9 POS Terminal

A point-of-sale (POS) terminal is a familiar sight at the end of a shopping trip. Create your own POS terminal to tally up the total price of barcoded items, and use external switches and an LCD display to make your terminal run as a stand-alone application.

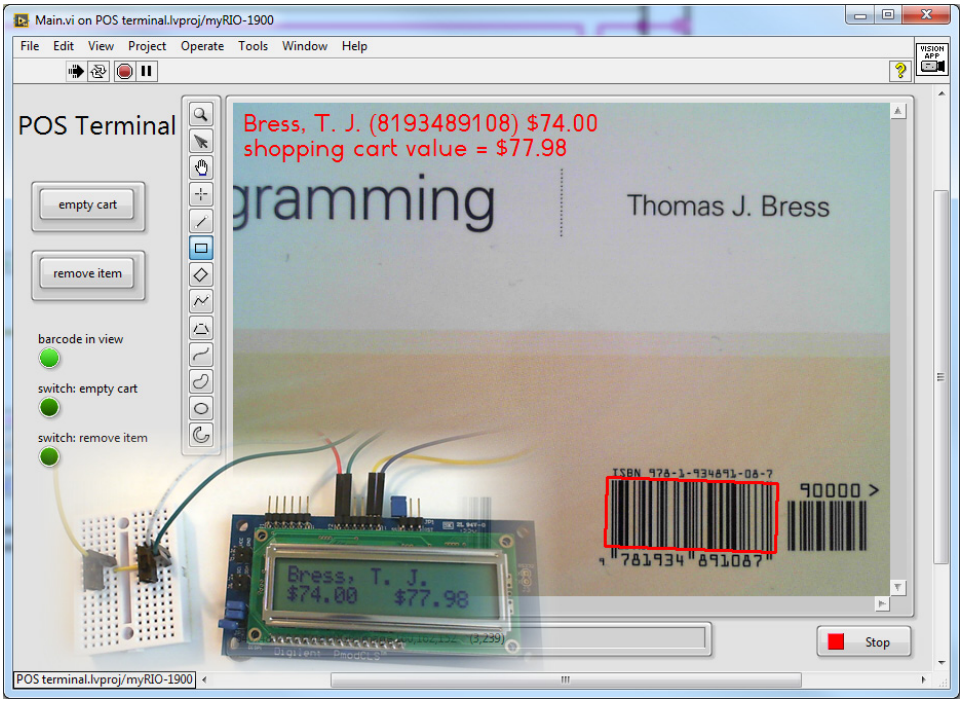

Figure 9.1: Front panel of the *POS Terminal* application.

Functional Requirements: The *POS Terminal* will:

1. Accept an image that contains a single EAN 13 barcode (equivalent to the UPC-A barcode),
2. Extract the 10-digit item code as a numerical value,
3. Look up the item's name and price from an inventory list,
4. Add the item price to the shopping cart total and subtract the item price when either the Remove Item front-panel button or an external switch is enabled,
5. Display the item's name, item code, item price, shopping cart value, and barcode bounding box as a nondestructive image overlay and on an external LCD display; display "unknown item" when a barcode is not found in the inventory list,
6. Activate two NI myRIO on-board LEDs and a front-panel indicator when a barcode is in view that has been read successfully,
7. Clear the shopping cart total to zero when either the Empty Cart front-panel button or an external switch is enabled, and
8. Display the state of the two external switches with two front-panel indicators and two NI myRIO on-board LEDs.

Required Resources:

1. NI LabVIEW and NI Vision Assistant
2. NI myRIO with power supply and USB cable
3. USB webcam and tabletop camera copy stand
4. LCD display (Digilent PmodCLS)
5. Two pushbutton or slide switches
6. Barcodes – any convenient items such as books or other products

Deliverables:

1. Lab report or notebook formatted according to instructor's requirements; include front-panel screen shots to demonstrate that your application meets functional specifications
2. .zip archive file containing all project files:

(a) Representative images
(b) Vision Assistant script
(c) LabVIEW project file set

9.1 Technical Approach

NI Vision Assistant Script

NI Vision Assistant provides the *Barcode Reader* step as a flexible and easy-to-use method to detect 1-D barcodes from among ten different standards including "EAN 13" (European Article Number) and the essentially equivalent "UPC-A" (Universal Product Code). This step can even detect multiple barcodes of different types, all within the same image. The *Barcode Reader* step is scale invariant (the barcode can be any size) as well as rotation invariant (the barcode angle does not matter); the only requirement is that the barcode must be in focus with sufficient spatial resolution to resolve the black and white bar pattern. Study *EAN-13 (UPC-A) Barcode Reader*[1] to learn how to set up the *Barcode Reader* step.

The Vision Assistant Express VI offers two indicators for the *Barcode Reader*; use the Barcodes output. This output is an array because the reader can read multiple barcodes at once. Use the following techniques to interpret the barcode output:

1. Use the Index Array function to retrieve a single barcode and connect this to a custom control in the data highway,

   > **TIP**: *Use the "Quick Drop" shortcut (Ctrl + space bar) to look up and place a LabVIEW element by name. See Appendix C on page 139 for more LabVIEW tips.*

2. The "Data" element of the barcode output is a string that contains the numerical code. Use the Empty String/Path? function to determine whether or not a valid barcode is in view, and
3. Use the Decimal String To Number function to convert the "Data" element string to an [I64] integer; wire a zero-valued constant of this data type to the `default` input.

[1] `http://youtu.be/hLSla3C-6V4` (1:29)

Digilent PmodCLS LCD Display

Chapters 26 to 28 of the *NI myRIO Project Essentials Guide*[2] provide a detailed explanation of the Digilent PmodCLS LCD display using its UART, SPI, or I^2C-bus interface as well as LabVIEW demonstration code available at *NI myRIO Project Essentials Guide .zip Files*[3] that you can drop into this project. Study *LCD Character Display Interfacing Theory*[4] to learn how to write escape sequences to configure and control the LCD display and then study one of these block diagram "walk-through" videos depending on the serial bus that you wish to use: *LCD (UART) Demo Walk-Through*[5], *LCD (SPI) Demo Walk-Through*[6], or *LCD (I2C) Demo Walk-Through*[7].

> **TIP**: *Refer to the NI myRIO connector diagrams in Appendix B on page 137.*

LabVIEW Techniques

Some tasks should execute only one time after a Boolean flag sets. For example, the `barcode in view` flag is clear when no valid barcode is visible and is set the entire time a valid barcode is visible. However, the `update shopping cart` task should execute one time *only* when the `barcode in view` flag first *changes* from clear to set. Study *Create "Flag Change" SubVI*[8] to learn how to create a subVI that detects when a flag changes state.

Use the Search 1D Array function to look up an item code in the inventory "item codes" array. This function returns the index of the matching element or -1 when no match is found. This index may then be used for the Index Array function.

Use the IMAQ Overlay Multiple Lines 2 VI to create the barcode bounding box from the "Bounding Box" element of the barcode Data custom control. As of this writing the VI throws an error that states that the bounding box does not contain enough points. However, simply disconnect its error cluster output and you should find that the overlay works properly anyways. Experiment with the `Width` and `Color` inputs to produce a satisfactory bounding box.

---

[2] `http://www.ni.com/myrio/project-guide`
[3] `http://www.ni.com/academic/myrio/project-guide-vis.zip`
[4] `http://youtu.be/m0Td7KbhvdI` (10:36)
[5] `http://youtu.be/JsEMMnIWg4k` (3:44)
[6] `http://youtu.be/oOXYryu4Y-c` (4:23)
[7] `http://youtu.be/qbD31AeqOMk` (4:32)
[8] `http://youtu.be/N1hUkWeZ8x0` (6:58)

These LabVIEW techniques are also required for this project:

- *Add Custom Data Lane*[9] – Create a custom control for the Barcodes Vision Assistant script output and insert this into the data highway
- *Create Formatted Text String*[10] – Format Into String
- *Overlay Text*[11] – Use IMAQ Overlay Text to insert a text string into the nondestructive image overlay
- *Select Overlay Text Font*[12] – Select the "NI Vision" font and adjust its size
- *Create "Inventory" Cluster of Arrays*[13] – Create a cluster of arrays to store the product inventory with item codes, names, and values

9.2 Development Tips

Representative Images

- Gather at least five items that contain barcodes.
- Conduct an Internet search to learn more about the structure of an EAN-13 barcode and write a paragraph to describe your findings. Also look for a site that can generate an EAN-13 barcode given a user-supplied value.
- Determine the camera distance and camera mode (resolution and frame rate) that achieves your required spatial resolution while maximizing the frame rate.
- Experiment with your barcode items to determine a suitable background color; you may find that a piece of white paper works well to protect the surface of the table-top camera copy stand. Ambient room light should suffice; keep the lighting as uniform as possible to avoid lighting "hot spots."
- Use a line grid target to align the camera so as to minimize perspective distortion.
- Adjust the camera attributes to maximize the contrast of the barcode's black-and-white pattern. The NI Vision Assistant *Barcode Reader* step is robust and should be able to ignore the other features visible on the object.

[9] `http://youtu.be/GkqjZCj3xck` (2:58)
[10] `http://youtu.be/VqH0SSK1fHY` (1:44)
[11] `http://youtu.be/SS72-mKX6fQ` (2:15)
[12] `http://youtu.be/lEKJcR553zA` (2:48)
[13] `http://youtu.be/DJB1otFHnbA` (5:03)

- Collect images of each of your barcoded items; use several different rotation angles (sideways, upside down, etc.) for each barcode.

9.3 Optional Extra Features:

You may wish to add one or more of these extra features:

- The NI Vision Assistant *Barcode Reader* step understands the "Code 39" barcode standard. You can easily create and print your own Code 39 barcodes with your favorite word processor and this free *IDAutomation.com Code-39 TrueType Font*[14]. Study *Code-39 Barcode Reader*[15] to learn how to set up the *Barcode Reader* step, and see additional technical details on this standard at *Code-39 Standard*[16].
- NI Vision Assistant also provides the *2D Barcode Reader* step that understands the popular QR code – see *QR Code Reader*[17] to get started. You can generate your own QR codes for free at *QR Code Generator*[18]. Even though a QR code normally encodes a website URL you can use any string that you like. Learn more about QR codes at *QR Code Standard*[19].
- The *OCR/OCV* step performs *optical character recognition* (OCR) to directly read printed text and numbers. The "OCR A Extended" font included with your word processor works quite well. See *OCR (Optical Character Recognition) Training*[20] to learn how to train the *OCR/OCV* step to read your own printed labels.
- The two switches in this project could be replaced with any two-state device presented in the *NI myRIO Project Essentials Guide* , e.g., the photodetector and photointerrupter.
- Add a buzzer to emit a short beep when the shopping cart updates (see Chapter 11 in the *NI myRIO Project Essentials Guide*).
- Calculate and add sales tax.
- Initialize the inventory list by reading a spreadsheet .csv file stored on the USB flash drive.

---

[14] `http://idautomation.com/free-barcode-products/code39-font`
[15] `http://youtu.be/iT8FAzkSksE` (2:29)
[16] `http://www.barcodeisland.com/code39.phtml`
[17] `http://youtu.be/8LTx9N0R8_g` (2:47)
[18] `http://www.qr-code-generator.com`
[19] `http://www.qrcode.com`
[20] `http://youtu.be/KxRSL3jnIAk` (5:04)

10 Keyed Optical Lock

Mechanical locks opened by metal keys represent a centuries-old technology. Modern-day electronic locks replace the metal key with encrypted radio pulses, RFID, magnetic stripes, and integrated circuits. Before the familiar metal keys disappear completely, try this "mash-up" of the old and the new by presenting a metal key to an optical scanner to make the decision as to whether or not to open a lock.

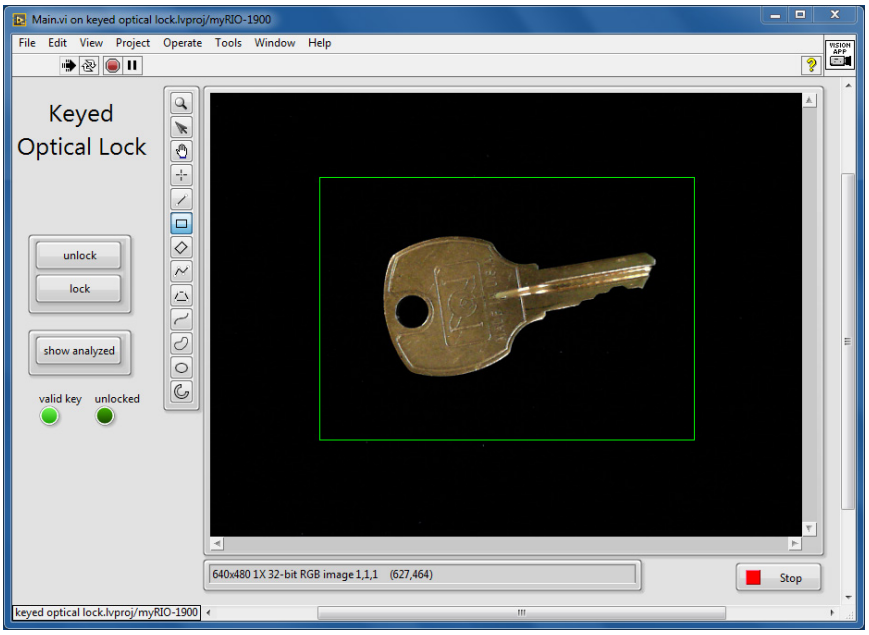

Figure 10.1: Front panel of the *Keyed Optical Lock* application.

Functional Requirements: The *Keyed Optical Lock* will:

1. Accept an image of a metal key (or multiple non touching keys),
2. Extract the perimeter contour of the key,
3. Compare the contour to a template stored on a USB flash drive,
4. Upon a successful match activate the following "lock devices" for a short time (e.g., 2 seconds): relay coil driven from a DIO line, the four on-board LEDs, and a front-panel indicator,
5. Wait for the key to be removed before enabling a new unlock-wait-lock cycle,
6. Indicate when a matching key is in view,
7. Manually control the lock devices with front-panel lock and unlock buttons, and
8. Display the analyzed image when show analyzed is pressed, otherwise display the webcam image.

Figure 10.1 on the preceding page shows the front-panel of the *Keyed Optical Lock* application and Figure 10.2 on the next page shows the image produced by the Vision Assistant script.

Required Resources:

1. NI LabVIEW and NI Vision Assistant
2. NI myRIO with power supply and USB cable
3. USB webcam and tabletop camera copy stand
4. Black velvet paper background
5. USB hub (Stratom X-HUB or equivalent)
6. USB flash drive
7. Relay and associated driver electronics; refer to Chapter 6 of the *NI myRIO Project Essentials Guide*[1] for a complete discussion of relays and sample LabVIEW code for this relay
8. Metal keys including two or more distinct keys cut from the same key blank style and two or more keys cut from other key blank styles

[1] `http://www.ni.com/myrio/project-guide`

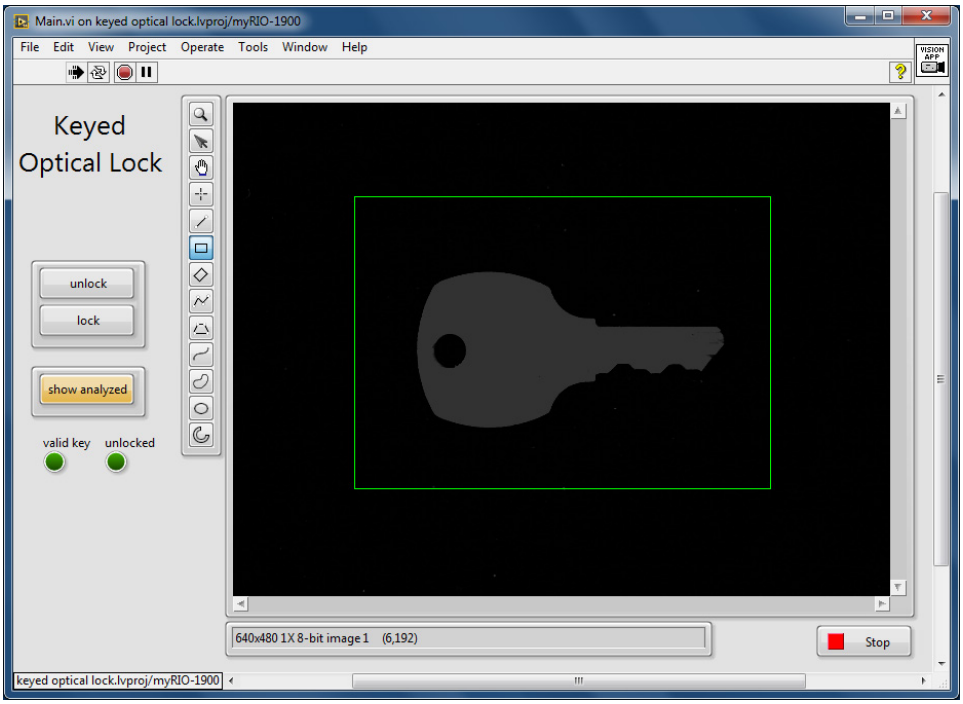

Figure 10.2: Front-panel for the *Keyed Optical Lock* application showing the output of the Vision Assistant script.

Deliverables:

1. Lab report or notebook formatted according to instructor's requirements; include front-panel screen shots to demonstrate that your application meets functional specifications
2. .zip archive file containing all project files:
   (a) Representative images
   (b) Vision Assistant script
   (c) LabVIEW project file set

10.1 Technical Approach

NI Vision Assistant Script

The NI Vision Assistant *Machine Vision | Geometric Matching* step extracts a boundary contour from a foreground object and compares this contour shape

to a stored template. Setting the "Minimum Score" for matching to a high value such as 990 out of 1000 ensures that the unique key cut pattern of a valid key will be distinguishable even from other keys cut from the same key blank. The *Geometric Matching* step provides rotation-invariant and scale-invariant matching. Flipping the key onto its other side yields a different contour, but you can place two *Geometric Matching* steps in the script to account for this possibility.

When incorporated into the Vision Assistant Express VI the *Geometric Matching* step offers the `Matches` output with complete details on the match results such as bounding box and match score; the `Number of Matches` numerical output is sufficient to indicate whether or not a valid key is present. The *Geometric Matching* step also accepts a user-selected ROI which you may wish to investigate as a means to reduce the CPU burden to search the entire field of view. See *Match Perimeter Contour*[2] to learn how to set up the *Geometric Matching* step to match perimeter contours against a template.

The reflected highlights from a metal key tend to distract the curve extraction routines employed by *Geometric Matching*. Clip the intensities to an upper limit with the *Grayscale | Operators* "Min" step; experiment with the intensity level to maintain the grayscale nature of the image while clipping any hot spots. See *Suppress Highlights*[3] to learn how to set up this step.

Relay from the "NI myRIO Starter Kit"

Chapter 6 of the *NI myRIO Project Essentials Guide*[4] provides a detailed explanation of the relay and associated driver electronics included with the "NI myRIO Starter Kit." Refer to the LabVIEW demonstration project in this chapter to obtain block diagram code that you can drop into this project. Study *Relay Interfacing Theory*[5] to learn how to interface the relay to one of the myRIO digital outputs, and study *Relay Demo Walk-Through*[6] to see the LabVIEW demonstration project in action.

LabVIEW Techniques

The following LabVIEW techniques are required for this project:

---

[2] `http://youtu.be/dLTFKucEM_A` (7:08)
[3] `http://youtu.be/XJaMjKxjLoo` (1:45)
[4] `http://www.ni.com/myrio/project-guide`
[5] `http://youtu.be/jLFL9_EWlwI` (11:11)
[6] `http://youtu.be/W2iukd8WVIA` (3:30)

- *Get ROI from Image Display*[7] – Use a Property Node to retrieve a rectangular ROI from the image display.
- Some tasks should execute only one time after a Boolean flag sets. For example, the `validkey` flag is clear when no valid key is visible and is set the entire time a valid key is in view. However, the `unlock-wait-lock` task should execute exactly one time *only* when the `validkey` flag first *changes* from clear to set. Study *Create "Flag Change" SubVI*[8] to learn how to create a subVI that detects when a flag changes state.

10.2 Development Tips

Representative Images

- Set up the camera on a table-top camera copy stand with an optically black background. Adjust the field of view to accommodate a single key or perhaps several keys. Allow for the key to be rotated to any angle. Set all of the camera attributes to manual, and adjust the brightness, contrast, and exposure to ensure that the background looks completely black while the foreground (metal key) maintains good contrast.
- Experiment with the available resolutions. Strive for a balance between good resolution on the key cuts and high frame rate.
- Select one of the keys as the "master key" and collect images of the key at various angles of rotation. Be sure to create one image in which the flat side of the key is horizontal, i.e., aligned with the image rows.
- Flip the key onto its back side and repeat. Again, ensure that one of the images shows the key's flat side as horizontal.
- Select another key that is of the same type but with a different cut pattern. Collect several images at various rotation angles.
- Select a key that is obviously of a different type and repeat.
- Optional: Collect several images of multiple keys in the field of view, some containing the master key and some not.

[7] `http://youtu.be/sKeO2iM8UWE` (2:29)
[8] `http://youtu.be/N1hUkWeZ8x0` (6:58)

Vision Assistant Script:

- The *Geometric Matching* step requires a grayscale image; see *Extract Luminance Plane*[9] to learn how to extract the luminance plane with *Color Plane Extraction*.
- Select the key image with the horizontal flat side on top and the key cuts on the bottom. Create a *Geometric Matching* step and then create a new template. Set the "Extraction Mode" to "Uniform Regions" because the background and foreground are both uniform in intensity. Your goal is a single closed contour around the periphery of the key; another contour for the hole in the key bow is acceptable. Click "OK" until you are prompted to save the template image; you will eventually need to copy this image to the USB thumb drive for myRIO.
- After you have created the template, select the "Settings" tab and retain only the "Rotated" option for matching. Preview all of your images and confirm that matches are found only for the various orientations of the master key and no other keys. Make note of the "Score" results, and then set the "Minimum Score" to reject any key that other than the master key.
- Create a second *Geometric Matching* step to match the master key when it is flipped to its back side. Figure 10.3 on the facing page illustrates typical expected script results.

[9] `http://youtu.be/1mmzsd46_Gs` (1:02)

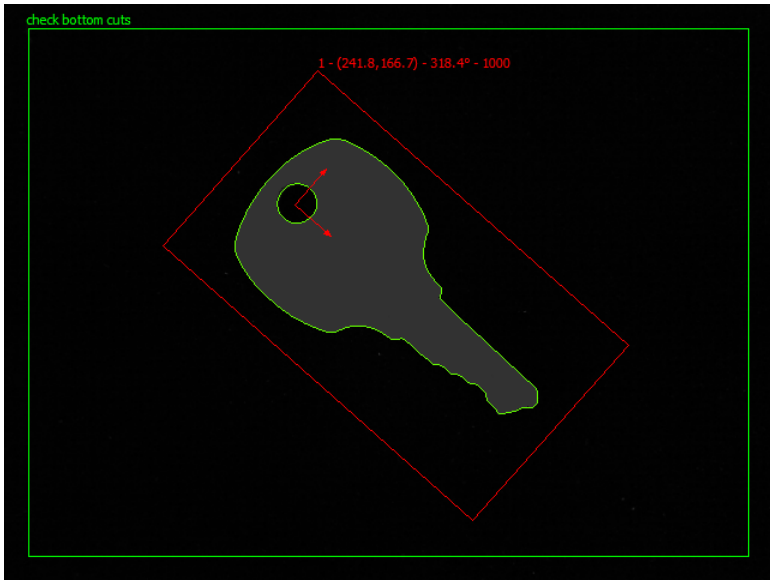

Figure 10.3: Typical expected results for the *Geometric Matching* step.

11 DMM Test Stand

Suppose you are in business to manufacture inexpensive digital multimeters (DMMs) that do not offer a digital data interface. As part of your product testing you need to apply a range of DC voltages and confirm that the numerical display is correct for each voltage. Create a test stand *for the DMM that will sweep the applied meter voltage, visually observe and record the resulting meter readings, and then plot the measured voltage as a function of the applied voltage.*

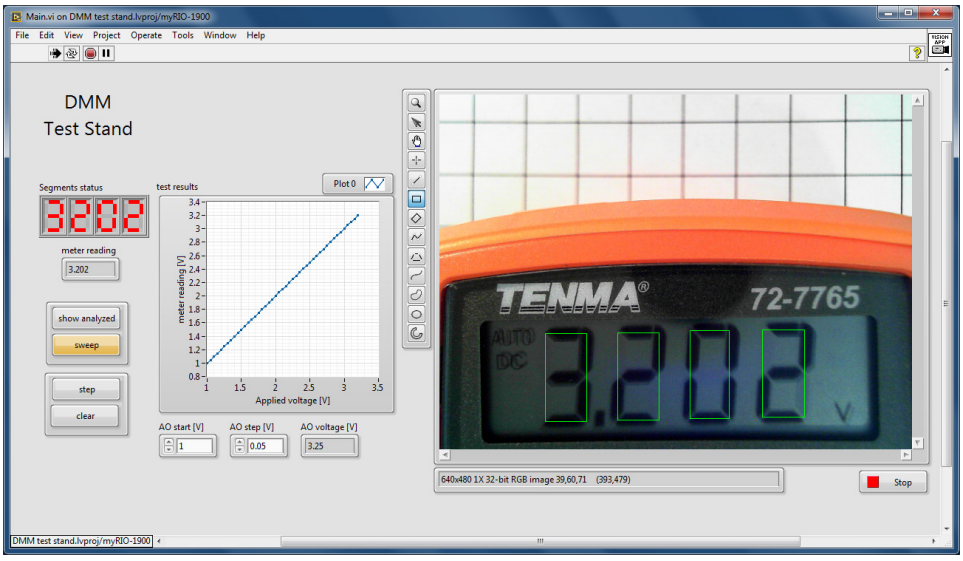

Figure 11.1: Front panel of the *DMM Test Stand* application.

Functional Requirements: The *DMM Test Stand* will:

1. Accept an image of the LCD or LED display of a DMM,
2. Display the image with a compound ROI overlay, one for each digit of the display,
3. Display the detected active segments,
4. Continually run the following measurement sequence while the sweep button is pressed:
   (a) Increment the analog output (AO) voltage by the user-supplied AO step value,
   (b) Wait for the meter display to stabilize,
   (c) Read the meter and save the measured voltage to an array, and
   (d) Update the plot of measured versus applied voltage.
5. Display the AO voltage applied to the meter and the numerical value read from the meter,
6. Clear the measurements and initialize the analog output voltage to the user-supplied AO start voltage when clear is pressed, and
7. Run a single measurement sequence when the step button is pressed.

Required Resources:

1. NI LabVIEW and NI Vision Assistant
2. NI myRIO with power supply and USB cable
3. USB webcam and tabletop camera copy stand
4. Hand-held DMM with LCD or LED display

Deliverables:

1. Lab report or notebook formatted according to instructor's requirements; include front-panel screen shots to demonstrate that your application meets functional specifications
2. .zip archive file containing all project files:
   (a) Representative images
   (b) Vision Assistant script
   (c) LabVIEW project file set

11.1 Technical Approach

NI Vision Assistant Script

NI Vision provides the IMAQ Read LCD instrument reader VI that accepts an image of a digital multimeter (DMM) display and interprets the pattern of active segments as a numerical value; it can also identify the decimal point in the display. IMAQ Read LCD works with either dark segments on a light background (LCD with passive illumination) or light segments on a dark background (backlit LCD, LED, and fluorescent).

The instrument reader is not available within Vision Assistant, therefore the script simply needs to convert the color webcam image to grayscale; see *Extract Luminance Plane*[1] for details.

The IMAQ Read LCD VI located in the Vision and Motion | Machine Vision | Instrument Readers subpalette requires a grayscale image buffer reference and a compound ROI (array of rectangular ROIs) with a single ROI for each digit. The companion VI IMAQ Get LCD ROI will automatically define the required multi-part ROI, however, this VI requires all segments to be active. Digital meters normally do this as part of their power-on self test, however, capturing this brief moment as an image is somewhat difficult and not worth effort. Fortunately you can easily create the compound ROI yourself on the main image display by drawing the first ROI as usual (right-click on the display and choose "Clear ROI," if necessary) and then press and hold the "Ctrl" button as you draw the remaining ROIs. The order in which you draw the ROIs dictates their position in the compound ROI array, therefore it is important that you begin with the leftmost digit and work your way to the right. See *Get Compound ROI from Image Display*[2] for details.

You will obtain best results when your meter display is aligned horizontally along the pixel rows and when the ROI boundaries fall in the *center* of the segments as shown in Figure 11.1 on page 81. You do *not* need to enclose the decimal points within the ROIs because the instrument reader knows where to look between your specified ROIs. You will need to experiment with the meter position to avoid the reflected image of the webcam itself (this is why the DMM is off center in Figure 11.1 on page 81). The instrument reader can handle a certain amount of lighting variation but will get confused if the background is not sufficiently uniform.

Study the video tutorial *Read a DMM Display*[3] to learn how to completely

[1] `http://youtu.be/1mmzsd46_Gs` (1:02)
[2] `http://youtu.be/a8UVbpmXxjM` (2:37)
[3] `http://youtu.be/b5YCZdhHMGM` (7:45)

configure the IMAQ Read LCD VI.

NI myRIO Analog Outputs

NI myRIO provides a total of eight analog outputs (AOs), with two on each of the two MXP connectors, two on the MSP connector, and two more on the audio output jack.

> **TIP**: *Refer to the NI myRIO connector diagrams in Appendix B on page 137.*

The MXP AOs produce 0 to 5 volts with 8-bit resolution while the MSP AO produces −10 to +10 volts with 12-bit resolution. See *myRIO Analog Output Express VI*[4] to learn how to place and configure the myRIO Analog Output Express VI.

LabVIEW Techniques

The following LabVIEW techniques are required for this project:

- *Get ROI from Image Display*[5] – Use a Property Node to retrieve a rectangular ROI from the image display.
- *Concatenate Scalar to an Array*[6] – Concatenate a scalar to an array
- *Create an X-Y Plot*[7] – Use a Bundle node to create an X-Y plot from two arrays.

11.2 Development Tips

- Adjust the camera attributes to maximize the contrast of the meter display. You may need to use manual focus to ensure that the segment edges are sharply defined.
- Use the Wait (ms) function to insert a delay immediately after you apply a new analog voltage to the meter to allow the meter reading to stabilize. Simply place this function in the same state in which you apply the meter voltage. Try values of approximately one second to get started, and then experiment to see if you can still get good measurement results with a shorter delay.

[4] `http://youtu.be/gaDlExM4WbU` (2:06)
[5] `http://youtu.be/sKeO2iM8UWE` (2:29)
[6] `http://youtu.be/l6HWwnL_9zI` (1:28)
[7] `http://youtu.be/TnKXqOsjyF0` (1:56)

> **TIP**: *Use the "Quick Drop" shortcut (Ctrl + space bar) to look up and place a LabVIEW element by name. See Appendix C on page 139 for more LabVIEW tips.*

11.3 Validation Tips

- Select "DC volts" mode on your DMM and connect it to your selected analog output,
- With no buttons pressed you should see the live webcam image, the active segments seen by the instrument reader, and the numerical value interpretation of the active segments,
- Define the compound ROI as described earlier,
- Click the show analyzed button and you should see the grayscale version of the live webcam image,
- Enter values for AO start and AO step, e.g., 1 volt and 0.05 volts, and then click the sweep button. Expect to see changes in the indicators for applied voltage, the meter reading, and the X-Y plot,
- Click the clear button and you should see the plot results cleared and the sweep restart from the AO start voltage,
- Click the sweep button again and the measurement should pause, and
- Click the step a few times and you should see an additional measurement point added with each button click.

Optional extra features:

You may wish to add one or more of these extra features:

- NI Vision provides an instrument reader for analog meters. If you have an analog DMM on hand you may find it interesting to create a machine vision application to "watch" the meter needle's position and turn this into a numerical value.

12 Gauging Station

Quality assurance for fabricated components requires gauging*, a procedure in which critical physical dimensions are measured and compared against specified tolerances. Traditional manual gauging requires tools such as dial calipers and micrometers, but you can create a vision-based gauging station that will take a large number of dimensional measurements simultaneously with the press of a button.*

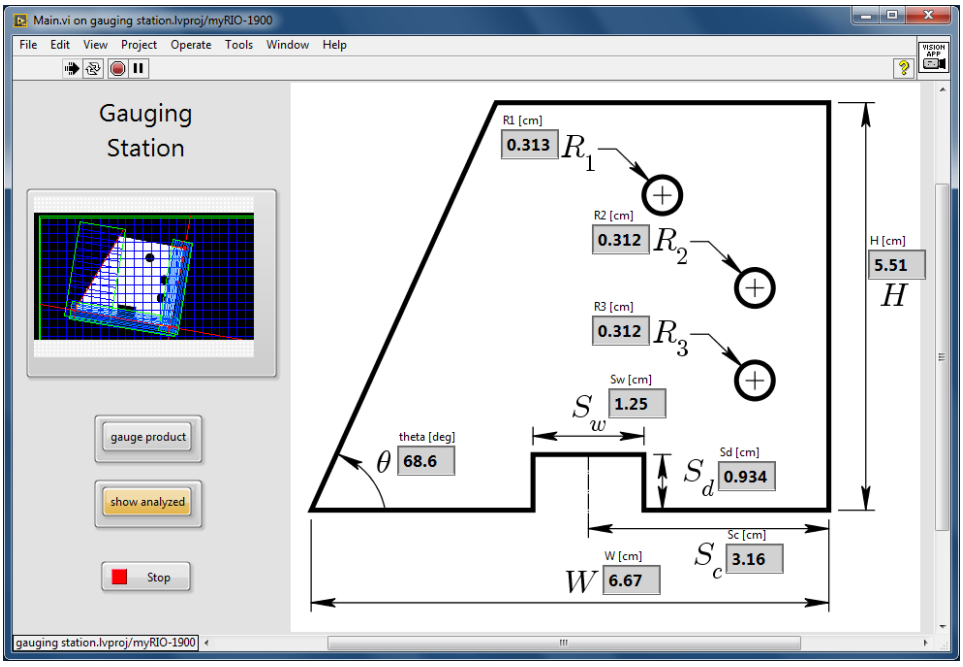

Figure 12.1: Front panel of the *Gauging Station* application.

Functional Requirements: The *Gauging Station* will:

1. Accept an image of a single component,
2. Apply the following gauging steps when the gauge button is pressed:
   (a) Apply grid calibration and correction to the image,
   (b) Locate a reference feature on the component and define a relative coordinate system based on this feature,
   (c) Take measurements on specified dimensions such as height, width, depth, angle, and hole radius, and
   (d) Report the measurements directly on an engineering drawing.
3. Display the analyzed image when show analyzed is pressed, otherwise display the webcam image.

Figure 12.1 on the previous page shows the front-panel of the *Gauging Station* application and Figure 12.2 shows the image produced by the Vision Assistant script.

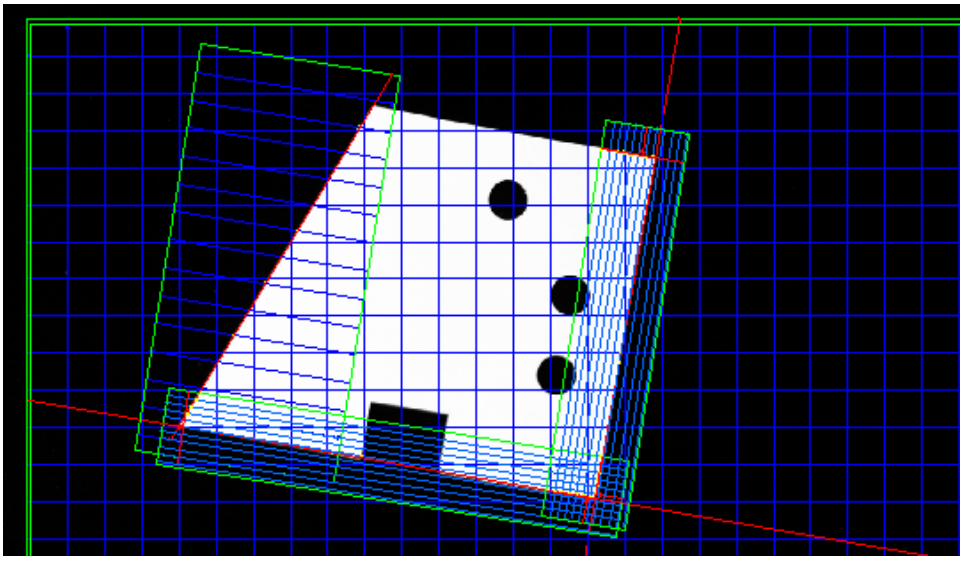

Figure 12.2: Analyzed image produced by the Vision Assistant script.

Required Resources:

1. NI LabVIEW and NI Vision Assistant

2. NI myRIO with power supply and USB cable
3. USB webcam and tabletop camera copy stand
4. Black velvet paper background
5. USB hub (Stratom X-HUB or equivalent)
6. USB flash drive
7. Paper cutout representation of a machined component

Deliverables:

1. Lab report or notebook formatted according to instructor's requirements; include front-panel screen shots to demonstrate that your application meets functional specifications
2. .zip archive file containing all project files:
   (a) Representative images
   (b) Vision Assistant script
   (c) LabVIEW project file set

12.1 Technical Approach

NI Vision Assistant Script

The NI Vision Assistant *Machine Vision* group contains a variety of tools to detect lines, edges, circles, and other features relevant to dimensional measurements. Figure 12.3 on the next page illustrates a suggested component that you can make with stiff paper, a scissors, and a hole punch to represent one side of a machined component that includes a bevel, a slot, and three bored holes. This component includes nine specified dimensions to be measured (all distances in centimeters):

1. H = component height,
2. W = component width,
3. S_d = slot depth,
4. S_w = slot width,
5. S_c = slot center as measured from the right edge,
6. θ = bevel angle in degrees, and
7. R_1,R_2,R_3 = three radii of the bored holes.

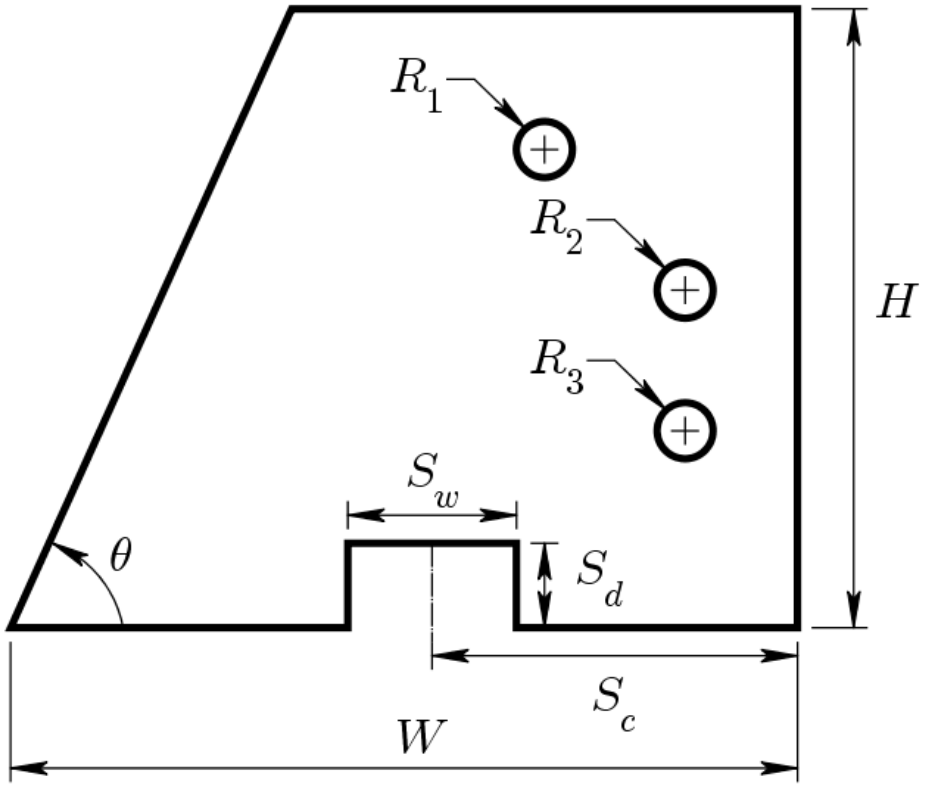

Figure 12.3: Component engineering drawing with specified measurements.

Calibrate and correct the image

The overall gauging process requires a relatively high resolution image that has been grid calibrated to minimize camera lens and perspective distortion; see *Grid Calibration*[1] for details. The calibration image must be stored on a USB flash drive to be accessible to the vision script running on NI myRIO. In addition, applying the correction to the image itself will improve the measurement accuracy and automatic detection of straight edges. *Correct a Grid-Calibrated Image*[2] illustrates this technique.

[1] `http://youtu.be/eWqq65ZZ0NQ` (6:47)
[2] `http://youtu.be/mJFsqqd37WY` (1:34)

Set the coordinate system to a reference feature

After calibrating and correcting the image the gauging application must identify a reference feature on the component to establish a local coordinate system for subsequent measurements. In this way the gauging application allows a certain degree of variety in component alignment with the camera pixel array and eliminates the need for a precise mechanical fixture to accomplish the alignment.

Use the following sequence of steps to place the origin of the coordinate system at the lower-right corner of the component pictured in Figure 12.3 on the facing page:

1. Find the vertical edge that corresponds to the right side of the component; see *Find Straight Edge*[3] to learn how to find a straight edge in a region of interest (ROI),
2. Find the horizontal edge that corresponds to the bottom of the component using another *Find Straight Edge* step,
3. Locate the intersection of these two lines as the bottom-right corner of the component to serve as the reference point for the coordinate system. See *Find the Intersection of Two Lines*[4] to learn how to use the *Caliper* step to find the coordinates of the intersection of these two lines, and
4. Define the coordinate system with its origin at the bottom-right corner of the component and aligned with the component's vertical edge; see *Set Coordinate System*[5] for details.

All ROIs that you define in subsequent script steps use this local coordinate system and will automatically follow the component as it translates and rotates.

Measure the component height and width

The *Clamp (Rake)* step easily measures H and W; see *Measure Width with "Clamp (Rake)"*[6] for details. If your representative image is not sufficiently aligned with the camera's pixel array you may need to use a rotated ROI as illustrated in *Measure Width with "Clamp (Rake)" (Rotated ROI)*[7].

> **IMPORTANT**: *Assign meaningful names to* every *step! You will frequently need to set up later steps based on the results of earlier steps, and the default names will rapidly become too confusing.*

[3] `http://youtu.be/eJ3KVGSlnuY` (2:25)
[4] `http://youtu.be/uOEhkFDuhVQ` (2:04)
[5] `http://youtu.be/PDcWiz2_mn0` (2:02)
[6] `http://youtu.be/fPWQCnB7B4s` (2:18)
[7] `http://youtu.be/bJawZuGBxsU` (2:36)

Measure the bored hole radii

Use three *Shape Detection* steps to look for each of the three circles. A single such step is able to detect all circles at once and present them as an array of measurement results, but then it is not easy to determine which measurement belongs to which circle. Study *Detect and Measure a Circular Shape*[8] to learn how to detect a circular shape and to measure its radius and center coordinates.

Find slot features and bevelled side

The *Edge Detector* finds edges along a straight-line ROI, for example, the two side-wall edges of the slot that defines the slot width S_w. See *Find Edge Locations*[9] to learn how to find the two side-wall edges. Use a *Caliper* step to locate the slot center as the midpoint between these two edges as illustrated by *Find Midpoint Between Two Points*[10]. Use another *Edge Detector* step to locate the slot bottom that partially defines the slot depth S_d.

Use a *Find Straight Edge* step to locate the bevelled left side edge of the component.

Take remaining measurements

The *Caliper* step can easily make many measurements at once and return them in a 2-D array as illustrated in *Multiple Caliper Measurements*[11]:

1. Bevel angle θ: Use the component's bottom and left edges to define the 4-point angle ("Angle 4 Points" geometric feature) as illustrated by *Measure the Angle Formed By Two Lines*[12],
2. Slot width S_w: Measure the distance between the two edge points found earlier ("Distance" geometric feature),
3. Slot depth S_d: Measure the "Perpendicular Projection" distance between the bottom edge of the component and the slot bottom edge as illustrated by *Measure a Perpendicular Projection Distance*[13], and
4. Slot center distance S_c: Measure the perpendicular projection distance between the right edge of the component and the slot midpoint found earlier.

[8] `http://youtu.be/nNhkbTFt3cU` (2:06)
[9] `http://youtu.be/6bcO-omuOi0` (3:21)
[10] `http://youtu.be/sTdAFjSBUXI` (1:55)
[11] `http://youtu.be/ZbGumGRaRqU` (3:13)
[12] `http://youtu.be/r57NBz8F69U` (2:06)
[13] `http://youtu.be/o6tNTR8Znhw` (2:22)

LabVIEW Techniques

Once your Vision Assistant script is packaged into an Express VI the LabVIEW project is almost done, however, interpreting and using the output of the *Caliper* step takes some care. See *"Gauging Demo" Connections*[14] to learn how to extract specific values from the 2-D array output of the *Caliper* step.

Take a screen capture of Figure 12.3 on page 90 and then paste this onto the front-panel of your application. Place frameless numerical indicators near the dimensional symbols as shown in Figure 12.1 on page 87 and then connect these indicators to your Vision Assistant Express VI.

12.2 Development Tips

Representative Images

- Set up the camera on a table-top camera copy stand with an optically black background. Adjust the field of view to accommodate a single component. Allow for the component to tilt and translate somewhat. Set all of the camera attributes to manual, and adjust the brightness, contrast, and exposure to ensure that the background looks completely black while the foreground component maintains good contrast.
- Use a relatively high resolution, e.g., 1280×720, but do not go too high, otherwise myRIO may not have enough memory to perform the needed grid calibration tasks. You could also convert the LabVIEW project to target the desktop and thereby eliminate any concerns about memory use.

  > **IMPORTANT**: *All of the ROIs that look for specific object features depend on a specific resolution, therefore you need to make this choice at the beginning of development. Changing the resolution after the Vision Assistant script is complete requires editing of nearly every step!*

- Collect one image in which the component bottom and right-side edges are aligned with the pixel array.
- Collect additional images in which the component is translated and tipped to the right and left.
- Make additional cutouts that contain "defects," e.g., incorrect bevel angle, slot width or depth, and hole radius.

[14] `http://youtu.be/auw8OBj_7m4` (6:54)

- If you plan to use the "image flip" option in the *Machine Vision App* template you will want to collect all of your images (including the grid calibration image) so that they appear in the same orientation when you develop your script. Consider one of these two options:
  - Run either the *Machine Vision App* project or one of your existing projects with the "image flip" option enabled, adjust the camera mode (resolution), right-click on the image display, and choose "Save Image,"
  - Run NI-MAX and save the images directly; these will need to be flipped by batch processing in Vision Assistant; see *Batch Process a Folder of Images*[15] for details. Manually renaming all of the processed images is a disadvantage of this technique, however.

[15] `http://youtu.be/xee96nDrUpw` (2:22)

13 Product Label Inspector

At first glance the artwork and graphics on a typical product label seem to pose a difficult inspection challenge – how can small imperfections such as tears or smudges be detected among so many features? The golden template *technique makes this type of inspection a snap: images are compared to a defect-free image (the golden template) and any difference regions are highlighted and counted.*

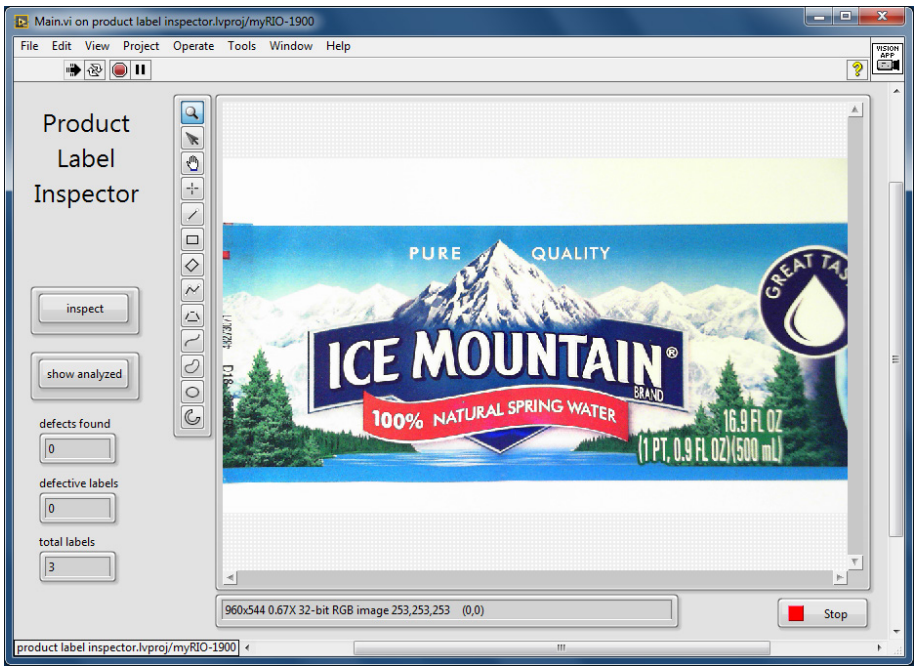

Figure 13.1: Front panel of the *Product Label Inspector* application.

Functional Requirements: The *Product Label Inspector* will:

1. Display the webcam image at the maximum possible frame rate while the product label is moved into position,
2. Compare the current image to the golden template image stored in the USB flash drive and report the number of defects found when the inspect button is pressed,
3. Accommodate arbitrary rotation and translation of the product label,
4. Illuminate a proportional number of NI myRIO on-board LEDs as a bar-graph to indicate the number of defects found,
5. Maintain a count of the total number of labels processed and a count of the number of defective labels found during the current session, and
6. Display the analyzed image when show analyzed is pressed, otherwise display the webcam image.

Figure 13.1 on the previous page shows the front-panel of the *Product Label Inspector* application viewing a defect-free product label that serves as the golden template. Figure 13.2 on the facing page shows another label with two pencil marks as defects and the corresponding binary image produced by the Vision Assistant analysis script.

Required Resources:

1. NI LabVIEW and NI Vision Assistant
2. NI myRIO with power supply and USB cable
3. USB webcam and tabletop camera copy stand
4. USB hub (Stratom X-HUB or equivalent)
5. USB flash drive
6. Paper product labels
7. Double-stick tape
8. Stiff paper cards

Deliverables:

1. Lab report or notebook formatted according to instructor's requirements; include front-panel screen shots to demonstrate that your application meets functional specifications

Figure 13.2: Product label with two pencil smudge defects (outlined) and the corresponding binary image produced by the analysis script.

2. .zip archive file containing all project files:

   (a) Representative images
   (b) Vision Assistant script
   (c) LabVIEW project file set

13.1 Technical Approach

NI Vision Assistant Script

The NI Vision Assistant *Machine Vision | Golden Template Comparison* step compares a region of interest (ROI) of an image to a template image and returns a binary image in which non-zero pixels indicate differences between the current image and the template. Conceptually the golden template method simply subtracts pixel-wise the template image from the current image and any significant difference in grayscale intensity indicates a defect. In practice, however, the subtraction will yield false positives due to even minor misalignment and lighting variations. The *Golden Template Comparison* step includes options to deal with misalignment by ignoring edges and to accommodate lighting variations by normalizing grayscale intensities with histogram matching. Figure 13.2 on the previous page illustrates the result of golden template comparison when the current image contains two pencil marks as defects; the red blobs or particles in the corresponding binary image indicate pixels that are darker than expected. Pixels that are brighter than expected would appear green. Study the tutorial video *Golden Template Comparison*[1] to learn how to set up this step and to create a golden template image.

The golden template comparison requires precise alignment between the candidate image and the template image, therefore the vision script must be able to adapt to the position of the label in the current image. The *Geometric Matching* step provides a convenient and effective way to search the image for a reference feature, and once found, define a coordinate reference based on this feature. In this way the product label may be presented with an unknown degree of translation and rotation, and then the script automatically establishes a local coordinate system that matches the way that the golden template image was defined. See *Find Reference Feature (Geometric Matching)*[2] to learn how to search the image for a reference feature, and then see *Set Coordinate System*[3] to learn how to set the coordinate system based on this feature.

The *Golden Template Comparison* step produces a binary image that will likely contain residual noise particles. See *Remove Small-Area Particles*[4] to learn how to use the *Particle Filter* step to remove these small particles by eliminating all particles smaller than a given area, and then see *Report Area of Particles*[5]

[1] `http://youtu.be/Ux7mQGaEmkI` (4:42)
[2] `http://youtu.be/o84e5EkLAs4` (5:40)
[3] `http://youtu.be/PDcWiz2_mn0` (2:02)
[4] `http://youtu.be/2v1sgSew8cs` (2:13)
[5] `http://youtu.be/N8T3HDdlN-E` (1:44)

to learn how to use the *Particle Analysis* step to count the remaining particles (now assumed to represent the product label defects) and to report their areas.

LabVIEW Techniques

The following LabVIEW techniques are required for this project:

- *(qsm) initialize: Front-Panel*[6] – Use an Invoke Node to reset the numerical front-panel indicators when the application starts up, and
- *Select Image Palette*[7] – Use a Property Node to select either the "Grayscale" or "Binary" palette for the image display; the *Particle Filter* step produces a binary output that appears like an all-black screen unless the "Binary" palette is selected.

13.2 Development Tips

Representative Images

- Use double-sided tape to bond the defect-free paper product label to a stiff paper card or piece of cardboard. The product label need to be as flat as possible when presented to the camera, and mounting the label to a card makes handling more convenient and prevents wrinkling.
- Create additional product label cards in which you introduce defects such as pencil mark smudges, tears, and holes of varying degrees of severity.
- Set up the camera on a table-top camera copy stand and adjust the field of view to accommodate a single product label. Allow for the label to be tilted and translated somewhat. Adjust the camera mode to obtain reasonably high resolution and the camera attributes to obtain good contrast without over or underexposure.

  > **IMPORTANT**: *Ensure that the camera's physical setup remains undisturbed throughout this project! The conditions under which you create the golden template image must be identical to the way that your finished vision application collects images of the product label.*

- Collect one image to serve as the basis of your golden template. The product label must be free of defects. Align the product label with the pixel array to the best of your ability.

[6] `http://youtu.be/eBtBy__PNwU` (1:49)
[7] `http://youtu.be/M53-QmPsSq4` (2:09)

- Collect additional images of the defect-free label with varying amounts of translation and rotation. Use these images during vision script development to verify that your script detects zero defects even when the label orientation changes.
- Collect multiple views of each product label defect, again to verify that your script delivers similar results for a given label defect with different orientations.
- If you plan to use the "image flip" option in the *Machine Vision App* template you will want to collect all of your images so that they appear in the same orientation when you develop your script. Consider one of these two options:
  - Run either the *Machine Vision App* project or one of your existing projects with the "image flip" option enabled, adjust the camera mode (resolution), right-click on the image display, and choose "Save Image,"
  - Run NI-MAX and save the images directly; these will need to be flipped by batch processing in Vision Assistant; see *Batch Process a Folder of Images*[8] for details. Manually renaming all of the processed images is a disadvantage of this technique, however.

[8] `http://youtu.be/xee96nDrUpw` (2:22)

14 Component Placement Inspector

Machine vision systems play an important role in the manufacture of printed circuit boards (PCBs). Just before the crucial reflow soldering step the board must be checked to ensure that all of the components are placed properly. Create a mock-up of a PCB component placement inspector that confirms whether or not a multitude of colored plastic bricks are properly attached to a baseboard.

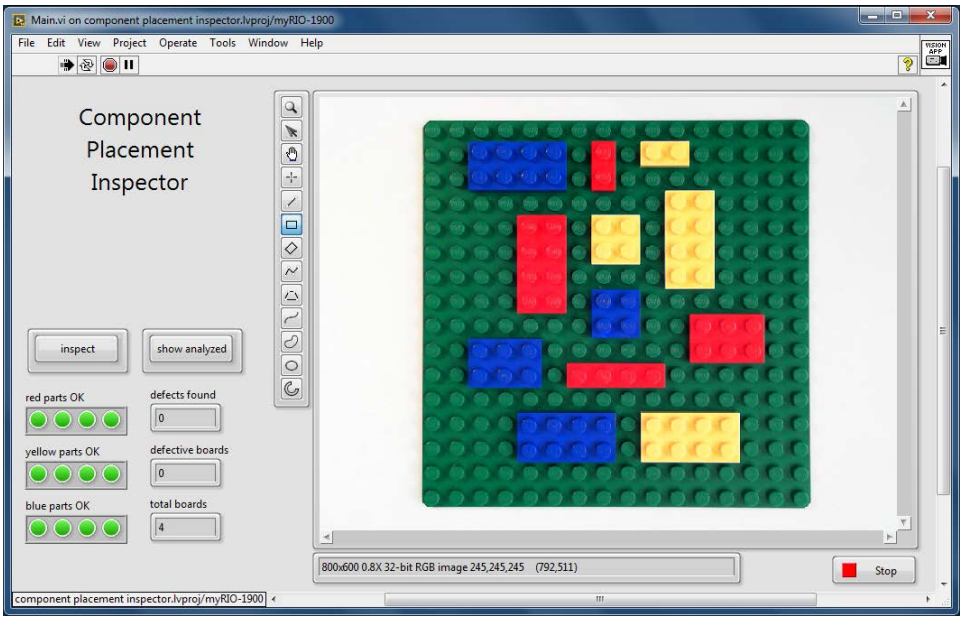

Figure 14.1: Front panel of the *Component Placement Inspector* application.

Functional Requirements: The *Component Placement Inspector* will:

1. Display the webcam image at the maximum possible frame rate while the board is moved into position,
2. Check each of 16 component sites for the correct color brick when the inspect button is pressed, and report the number of defects found (incorrectly placed or missing components),
3. Indicate the pass/fail status of each component,
4. Accommodate arbitrary rotation and translation of the board,
5. Illuminate a proportional number of NI myRIO on-board LEDs as a bargraph to indicate the number of defects found,
6. Maintain a count of the total number of boards processed and a count of the number of defective boards found during the current session, and
7. Display the analyzed image when show analyzed is pressed, otherwise display the webcam image.

Figure 14.1 on the preceding page shows the front-panel of the *Component Placement Inspector* application viewing a defect-free board. Figure 14.2 on the next page shows the analyzed image when the board is tilted and missing a red brick and a yellow brick.

Required Resources:

1. NI LabVIEW and NI Vision Assistant
2. NI myRIO with power supply and USB cable
3. USB webcam and tabletop camera copy stand
4. Green plastic baseboard
5. Four each of yellow, red, and blue thin plastic bricks

Deliverables:

1. Lab report or notebook formatted according to instructor's requirements; include front-panel screen shots to demonstrate that your application meets functional specifications
2. .zip archive file containing all project files:
   (a) Representative images
   (b) Vision Assistant script
   (c) LabVIEW project file set

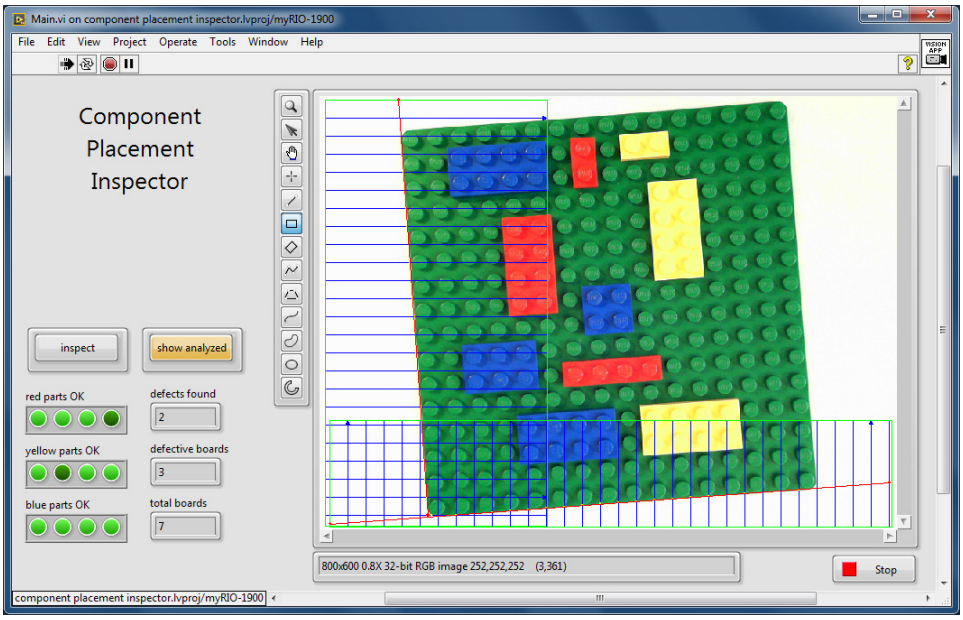

Figure 14.2: Analyzed output from the vision script with a board that is tilted and missing a red brick and a yellow brick.

14.1 Technical Approach

NI Vision Assistant Script

The NI Vision Assistant *Color Matching* step compares a region of interest (ROI) to a color and indicates whether or not the ROI matches the expected color. This step can accommodate a compound ROI, therefore multiple sites can be checked within a single script step. When packaged in a Vision Assistant Express VI the *Color Matching* step can generate both a Match Flag (Boolean pass/fail indicator) and a Match Score (numerical value between 0 and 1000). Study the tutorial video *Check for Matching Color*[1] to learn how to set up the *Color Matching* step to check multiple component sites for an expected color.

> **NOTE**: *The ROIs defined in the Color Matching step depend on a specific camera resolution, therefore, settle on a resolution early in your development process to avoid having to redefine the color matching ROIs.*

[1] `http://youtu.be/VmO5fBoZMyE` (3:23)

The *Color Matching* ROIs must be placed relative to a coordinate system derived from a reference feature on the board to permit the vision script to adapt to some baseboard rotation and translation. Use the following sequence of steps to place the origin of the coordinate system at the lower-left corner of the board:

1. Find the vertical edge that corresponds to the left side of the board; see *Find Straight Edge*[2] to learn how to find a straight edge in a region of interest (ROI),
2. Find the horizontal edge that corresponds to the bottom of the board using another *Find Straight Edge* step,
3. Locate the intersection of these two lines as the bottom-left corner of the board to serve as the reference point for the coordinate system. See *Find the Intersection of Two Lines*[3] to learn how to use the *Caliper* step to find the coordinates of the intersection of these two lines, and
4. Define the local coordinate system with its origin at the bottom-left corner of the board and aligned with the board's bottom edge; see *Set Coordinate System*[4] for details.

All ROIs you that you place in subsequent script steps reside in this local coordinate system and will automatically follow the board as it translates and rotates.

LabVIEW Techniques

The following LabVIEW techniques are required for this project:

- *(qsm) initialize: Front-Panel*[5] – Use an Invoke Node to reset the numerical front-panel indicators when the application starts up.
- The Match Flag indicator produced by the *Color Matching* step is a Boolean array. Create an indicator directly from this array output to show the inspection results for all of the components of a given color. A correctly-placed part generates a "true" value.
- Use a Boolean Not function wired directly to the Match Flag array to make placement errors appear as "true" values, convert this array to an integer array with Boolean To (0,1) (the true/false values now appear as 0/1 values), and then sum the array elements to a single numerical value

[2] http://youtu.be/eJ3KVGSlnuY (2:25)
[3] http://youtu.be/uOEhkFDuhVQ (2:04)
[4] http://youtu.be/PDcWiz2_mn0 (2:02)
[5] http://youtu.be/eBtBy__PNwU (1:49)

with Add Array Elements that indicates the number of defects found for a given component color.

> **TIP**: *Use the "Quick Drop" shortcut (Ctrl + space bar) to look up and place a LabVIEW element by name. See Appendix C on page 139 for more LabVIEW tips.*

14.2 Development Tips

Representative Images

- Create a mock-up of a printed circuit board with properly-placed components similar to that pictured in Figure 14.1 on page 101. Use several of each color and size of brick.
- Set up the camera on a table-top camera copy stand and adjust the field of view to accommodate a single board. Allow for the board to be tilted and translated somewhat. Adjust the camera mode to obtain reasonably high resolution and the camera attributes to obtain good contrast without over or underexposure.

  > **NOTE**: *Ensure that the camera's physical setup remains undisturbed throughout this project! The conditions under which you define the color matching ROIs must be identical to the way that your finished vision application collects images of the board.*

- Collect one image to serve as the basis of your color matching ROIs. Align the board with the pixel array to the best of your ability.
- Collect additional images of the defect-free board with varying amounts of translation and rotation. Use these images during vision script development to verify that your script detects zero defects even when the board orientation changes.
- Collect multiple views of each product label defect category, including missing bricks, swapped bricks, etc.
- If you plan to use the "image flip" option in the *Machine Vision App* template you will want to collect all of your images so that they appear in the same orientation when you develop your script. Consider one of these two options:
  - Run either the *Machine Vision App* project or one of your existing projects with the "image flip" option enabled, adjust the camera

mode (resolution), right-click on the image display, and choose "Save Image,"

– Run NI-MAX and save the images directly; these will need to be flipped by batch processing in Vision Assistant; see *Batch Process a Folder of Images*[6] for details. Manually renaming all of the processed images is a disadvantage of this technique, however.

[6] `http://youtu.be/xee96nDrUpw` (2:22)

15 Motion Detector

Do you have some property in a room that you would like to secure? Set up your webcam as a motion detector that will sound an alarm the moment that even the smallest feature in the camera's field of view has noticeable motion, also known as optical flow *velocity.*

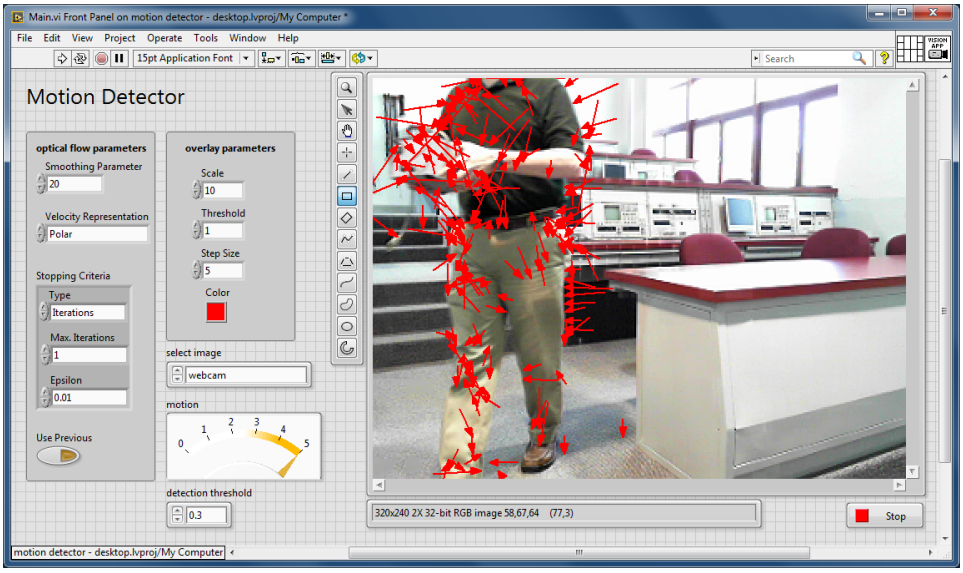

Figure 15.1: Front panel of the *Motion Detector* application.

Functional Requirements: The *Motion Detector* will:

1. Compute the optical flow of the imaged scene using the LabVIEW IMAQ Optical Flow (HS) and IMAQ Overlay Motion Vectors VIs,
2. Provide front-panel controls for both of the optical flow VIs,
3. Display any of the following user-selectable images:
   (a) Original color webcam image with an overlay of the optical flow vector field,
   (b) Current image frame in grayscale format,
   (c) Previous image frame in grayscale format,
   (d) Velocity Image #1 computed from the optical flow VI,
   (e) Velocity Image #2 computed from the optical flow VI, and
   (f) Thresholded version of the Velocity Image #1
4. Display the motion activity with a panel meter,
5. Activate all four NI myRIO on-board LEDs when the motion activity value exceeds a user-defined threshold.

Figure 15.1 on the preceding page shows the front-panel of the *Motion Detector* application viewing a moving object. The arrows indicate optical flow velocity throughout the image.

Required Resources:

1. NI LabVIEW and NI Vision Assistant
2. NI myRIO with power supply and USB cable
3. USB webcam and tabletop camera copy stand

Deliverables:

1. Lab report or notebook formatted according to instructor's requirements; include front-panel screen shots to demonstrate that your application meets functional specifications
2. .zip archive file containing all project files:
   (a) Representative images
   (b) Vision Assistant script
   (c) LabVIEW project file set

15.1 Technical Approach

NI Vision Assistant Script

The vision script merely needs to extract the luminance image from the color webcam image because the optical flow VIs are not available in Vision Assistant; see *Extract Luminance Plane*[1] for details.

Optical Flow VIs

The LabVIEW IMAQ Optical Flow (HS) VI computes the optical flow (velocity flow) from two successive grayscale image frames using the Horn and Schunck algorithm. Optical flow is a vector field, therefore the VI calculates two single-precision floating point images ([SGL]) in either Cartesian or polar form. The two images are generically called Velocity Component 1 and Velocity Component 2, and contain the X and Y vector components with "Cartesian" velocity representation selected and the magnitude and phase with "Polar" representation selected. The VI includes additional inputs such as stopping criteria, smoothing parameter, and the option to use the previously-calculated output as a starting point for the new calculation.

The IMAQ Overlay Motion Vectors VI accepts the pair of velocity component images generated above and generates a vector field overlay for a suitable image, i.e., the original webcam image. This VI must use the same selected velocity representation as that calculated by IMAQ Optical Flow (HS), of course, and the VI provides the ability to select the velocity threshold at which flow vectors are drawn, the length of the arrows (scale) as well as their color, and a step size to set the density of arrows.

LabVIEW Techniques

The following LabVIEW techniques are required for this project:

- *(qsm) initialize: Image Buffer*[2] – Use the IMAQ Create VI to create image buffers for the previous webcam frame ([Grayscale (U8)]), the two velocity components ([Grayscale (SGL)]), and the detected motion (also [Grayscale (SGL)]). Place these VIs in the ***(qsm) initialize*** state.
- The usual "mode" and "action" buttons are not required for this project. Instead of deleting the controls, simply right-click them in the block di-

[1] `http://youtu.be/1mmzsd46_Gs` (1:02)
[2] `http://youtu.be/suCkeB_afOc` (2:02)

agram and select "Hide Control." This way you can easily unhide them later should they be needed again.

- Use the IMAQ Copy VI in the ***(image) analyze*** state to copy the existing `qsm.Analyzed image` buffer contents to the `previous image` buffer. Be certain to execute this VI *before* the Vision Assistant script using error cluster propagation to control data flow, otherwise your copied image will always be identical to the current webcam image.
- Create an enumerated selector and a case structure to determine which of the six images to show on the main image display in the ***(image) show*** state.
- Use the IMAQ Quantify 2 VI to determine the maximum and minimum values in the Velocity Component 1 image, and then use the difference of these two values as a measure of motion activity to be displayed by the panel meter.

  > **TIP**: *Use the "Quick Drop" shortcut (Ctrl + space bar) to look up and place a LabVIEW element by name. See Appendix C on page 139 for more LabVIEW tips.*

- The IMAQ Optical Flow (HS) VI will throw an error if the current frame and previous frame do not each contain valid images of the same size. Your application will need to enqueue a special task to select the ***(image) analyze*** state one time to create the "Analyzed" image before the default task is selected. In this way the previous image frame will be available the first time that the optical flow VI is called. Use the First Call? function to detect the very first time that the scheduler runs.

Optional extra features:

You may wish to add one or more of these extra features:

- Add an audible alarm; see Chapter 11 of the *NI myRIO Project Essentials Guide*[3] for details on the piezoelectric buzzer.
- Add a time stamp to the image overlay.
- While motion is detected save an image to the USB thumb drive once every second or so.

[3] `http://www.ni.com/myrio/project-guide`

16 Auto-Pan Camera

Suppose you need to make a video recording of a speaker's presentation. The speaker usually stands at a podium, but occasionally moves side to side and may even walk away from the podium altogether. Create an auto-pan camera *that combines a servo-mounted webcam, a position sensing algorithm based on color, and a closed-loop control system move the camera so as to maintain the target object in the center at the camera's field of view.*

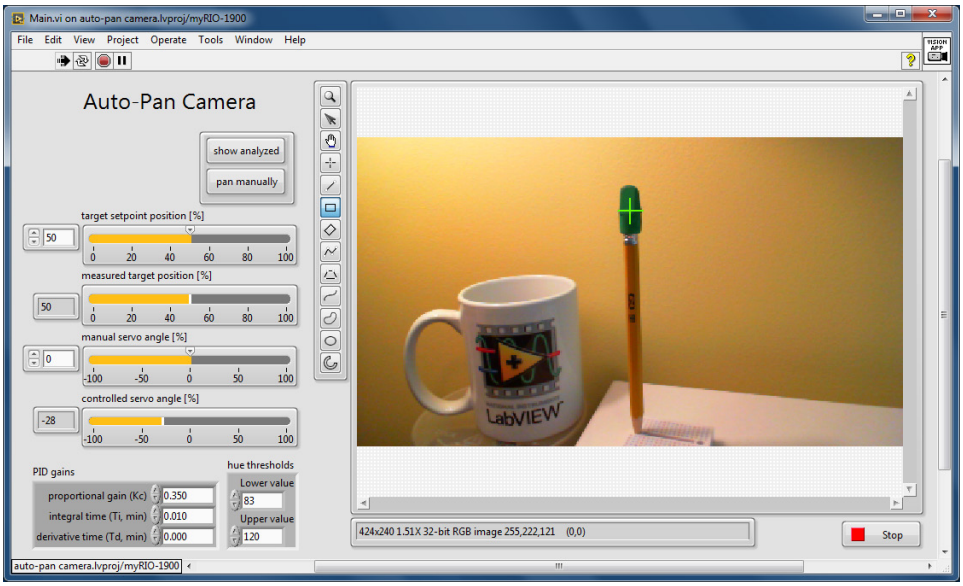

Figure 16.1: Front panel of the *Auto-Pan Camera* application.

Functional Requirements: The *Auto-Pan Camera* will:

1. Identify the centroid coordinates of a target object based on its hue and user-supplied threshold values,
2. Display the measured target position as a percentage between 0% and +100% (left and right image boundaries),
3. Adjust the angle of a servo-mounted webcam to maintain the target at a user-defined set point position between 0% and 100% ,
4. Use a PI (proportional-integral) controller tuned for smooth tracking with minimal oscillation and overshoot with user-supplied gain values,
5. Display the controlled servo angle as a percentage between −100% and +100% (full left and full right),
6. Provide user control of the servo angle between −100% and +100% when pan manually is pressed,
7. Display the analyzed image when show analyzed is pressed, otherwise display the webcam image, and
8. Place a cross-hair centered at the measured target position in both the webcam and analyzed images.

Figure 16.1 on the preceding page shows the front-panel of the *Auto-Pan Camera* application viewing a pencil eraser as the tracked object. Figure 16.2 on the next page shows the corresponding image produced by the Vision Assistant script.

Required Resources:

1. NI LabVIEW and NI Vision Assistant
2. NI myRIO with power supply and USB cable
3. USB webcam and tabletop camera copy stand
4. Servo motor (*GWS S03N STD*[1]); refer to Chapter 17 of the *NI myRIO Project Essentials Guide*[2] for a complete discussion of servo motors and sample LabVIEW code for this servo
5. Duct tape

[1] http://gwsus.com/english/product/servo/standard.htm
[2] http://www.ni.com/myrio/project-guide

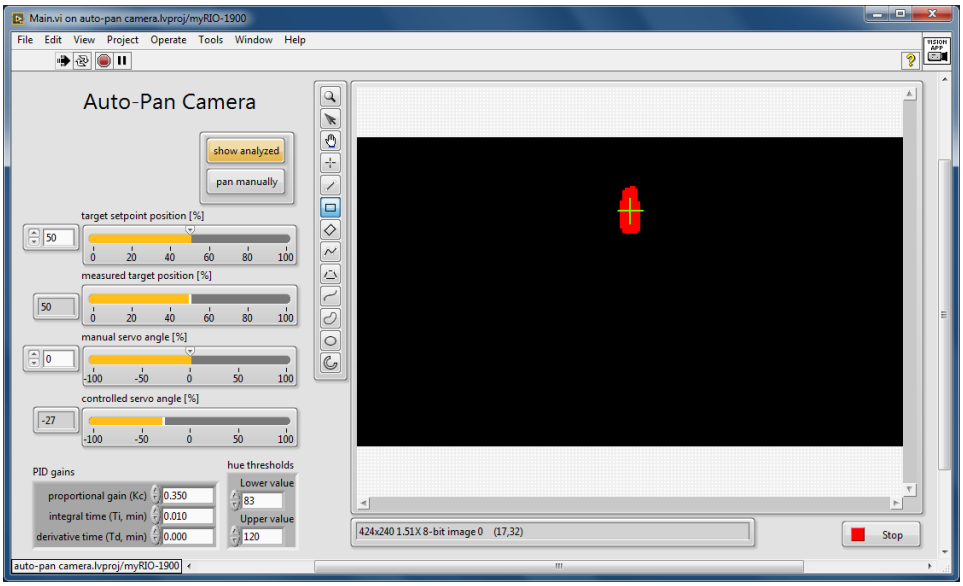

Figure 16.2: Front-panel for the *Auto-Pan Camera* application showing the output of the Vision Assistant script.

Deliverables:

1. Lab report or notebook formatted according to instructor's requirements; include front-panel screen shots to demonstrate that your application meets functional specifications
2. .zip archive file containing all project files:
   (a) Representative images
   (b) Vision Assistant script
   (c) LabVIEW project file set

16.1 Technical Approach

NI Vision Assistant Script

Broadly speaking, the vision script will accept a color image and return the centroid coordinates of an object that matches a defined color and the number of detected objects. To accomplish this task the vision script will extract the hue

plane (refer to *Extract Luminance Plane*[3] but select hue instead of luminance, of course), dual-threshold the hue image with user-defined low and high values to select the pixels that match the defined hue (see *Dual-Threshold Grayscale to Binary*[4]), remove noise pixels and other small objects (see *Remove Small Objects*[5]), find the energy center (centroid) of the image (see *Find Energy Center (Centroid)*[6]), and report the number of detected objects (see *Report Area of Particles*[7] this step serves *only* to count the particles, therefore choose any single measurement such as "Center of Mass X").

Select the thresholding `Range` control and the `Centroid` and `Number of Particles` indicators when you import this script into the Vision Assistant Express VI. The hue thresholds will likely need adjustment occasionally as lighting conditions change.

Camera Position Controller

The webcam is mounted to a servomotor as shown in Figure 16.3 on the facing page; the servomotor pans the camera to any desired angle over a 180° range.

The camera position controller can be completely implemented in the ***(myRIO) update ports*** state. In this state the tracked object's measured X-position from the vision script is converted into a value between 0 and 100, and the PID Advanced VI compares this measured position to the desired position to generate a command signal to the servo via the PWM Express VI; the command signal manipulates the servomotor to minimize the difference (error signal) between the measured position and the desired position.

> **TIP**: *Use the "Quick Drop" shortcut (Ctrl + space bar) to look up and place a LabVIEW element by name. See Appendix C on page 139 for more LabVIEW tips.*

Measured X-position

The vision script produces two outputs, the tracked object's position and the number of objects detected. Enclose the PID and PWM VIs and associated code in a Boolean case structure so that these run *only* when a single tracked object is detected. This way the camera will remain stationary when the tracked object disappears and will also not be distracted by competing features.

---

[3] `http://youtu.be/1mmzsd46_Gs` (1:02)
[4] `http://youtu.be/0TFh4zB6e3Y` (1:56)
[5] `http://youtu.be/Xfo9wK_4B7Q` (1:09)
[6] `http://youtu.be/XkYfR141x0k` (0:57)
[7] `http://youtu.be/N8T3HDdlN-E` (1:44)

Figure 16.3: Webcam mounted to the servomotor which in turn is mounted to a small tripod. Use strips of duct tape to secure the camera and servomotor to the tripod head.

Convert the tracked object's X-position from pixels into a [DBL] value between 0 (left image boundary) and 100 (right image boundary). Use IMAQ Get Image Size to determine the number of image pixels in the X-direction instead of hardcoding this value.

Servo control

Chapter 17 of the *NI myRIO Project Essentials Guide*[8] describes how to set the angle of a servomotor with the NI myRIO PWM (pulse-width modulated) output; see *Servo Interfacing Theory*[9] for details. Use the block diagram code described in that chapter to convert a desired angle expressed in percent of full-scale rotation (-100 to $+100$) into the corresponding duty cycle of the PWM

[8] `http://www.ni.com/myrio/project-guide`
[9] `http://youtu.be/DOu5AvSDP2E` (7:18)

output; see *Servo Demo Walk-Through*[10] to learn more about the code which may be downloaded at *NI myRIO Project Essentials Guide .zip Files*[11].

The demo code uses the low-level PWM VIs Open and Set Duty Cycle and Frequency. For the *Auto-Pan Camera* project it is much easier to reuse only the code that converts percent of full-scale rotation into duty cycle and then use this value as the input to the high-level PWM Express VI. You only need a single instance of this Express VI.

> **TIP**: *Refer to the NI myRIO connector diagrams in Appendix B on page 137.*

PID Controller

The closed-loop PID controller can be easily implemented with the PID Advanced VI located in the "Control & Simulation" subpalette. Connect to the following inputs:

- `setpoint` – Create a front-panel control to set the desired position of the tracked object to a value between 0 (left image boundary) and 100 (right image boundary). Use a default value of 50 to place the object in the center of the field of view,
- `process variable` – Connect to the measured X-position, a value between 0 and 100,
- `manual control` – Create a front-panel control to set the desired angle of the servomotor when the PID controller is set to manual mode. Use a range of -100 to $+100$,
- `auto?` – Use a logic inverter between this terminal and the pan manually front-panel button,
- `PID gains` – Create a front-panel control and set the values to those shown in Figure 16.1 on page 111, and
- `output` – Connect this signal to the PWM conversion code; this output ranges from -100 to $+100$.

Feel free to experiment with the PID gains. For example, reduce the proportional gain K_C if the system oscillates, and decrease the integral time T_i if

[10] `http://youtu.be/QXHe0DFbUdc` (4:23)
[11] `http://www.ni.com/academic/myrio/project-guide-vis.zip`

the camera takes too long to settle on the target object after it has moved to a new position. Refer to these articles on PID control loop tuning at Motion Engineering's page *Servo Tuning*[12] to learn more.

LabVIEW Techniques

The following LabVIEW techniques are required for this project:

- *Add Custom Data Lane*[13] – Add a custom lane: create a control from the Vision Assistant Express VI indicator output, convert to type def, save type def as a control, add the control to the data highway, merge the Vision Assistant Express VI output onto the data highway, and then retrieve the value in another state.
- *Overlay Line*[14] – Create a nondestructive line overlay with the "IMAQ Overlay Line" VI. Use two instances of this VI to draw the two cross-hair lines centered at the target's position.
- *Math with "Position" Clusters*[15] – Perform basic mathematical operations with the "Position" cluster used by NI Vision tools.
- *Select Image Palette*[16] – Use a Property Node to set the image display palette to either "Binary" or "Grayscale." The vision script produces a binary output that appears like an all-black screen unless the "Binary" palette is selected.

16.2 Development Tips

- Choose a target object that has a unique color, is relatively small compared to the field of view, and can be easily repositioned. A pencil with a colored eraser works well for this purpose.
- Use a relatively low camera resolution to ensure that the effective sampling rate for the control loop is as fast as possible.
- Use the "Manual" setting for the "White Balance" camera attribute. The default "Auto" setting can often shift the color of the tracked object outside your hue threshold values.

[12] `http://support.motioneng.com/downloads-notes/tuning/default.htm`
[13] `http://youtu.be/GkqjZCj3xck` (2:58)
[14] `http://youtu.be/amunze4umlg` (2:30)
[15] `http://youtu.be/15V-gJwRI2Y` (3:26)
[16] `http://youtu.be/M53-QmPsSq4` (2:09)

- Get some inspiration for extreme-performance auto-panning developed by the Ishikawa Watanabe Laboratory of the University of Tokyo. Two demo videos show the *1ms Auto Pan-Tilt*[17] system in action as it tracks a ping-pong ball in play as well as a yo-yo in the hands of an expert. The tracked object remains perfectly centered in the camera's field of view at all times!

Optional extra features:

You may wish to add one or more of these extra features:

- Use motion sensing as the basis for target detection (see Chapter 15 on page 107).
- Illuminate on-board LEDs to indicate status such as target object in view, camera in motion, etc.

[17] `http://www.k2.t.u-tokyo.ac.jp/mvf/SaccadeMirrorFullHD/index-e.html`

17 Marble Sorter

Vision systems excel at sorting objects into categories according to size, shape, and color. A typical sorting system consists of a hopper, feeder, camera, vision algorithm, and selector. Create a marble sorter based on a color camera and two servomotors that automatically sorts a hopper full of marbles into bins, one for each color of marble.

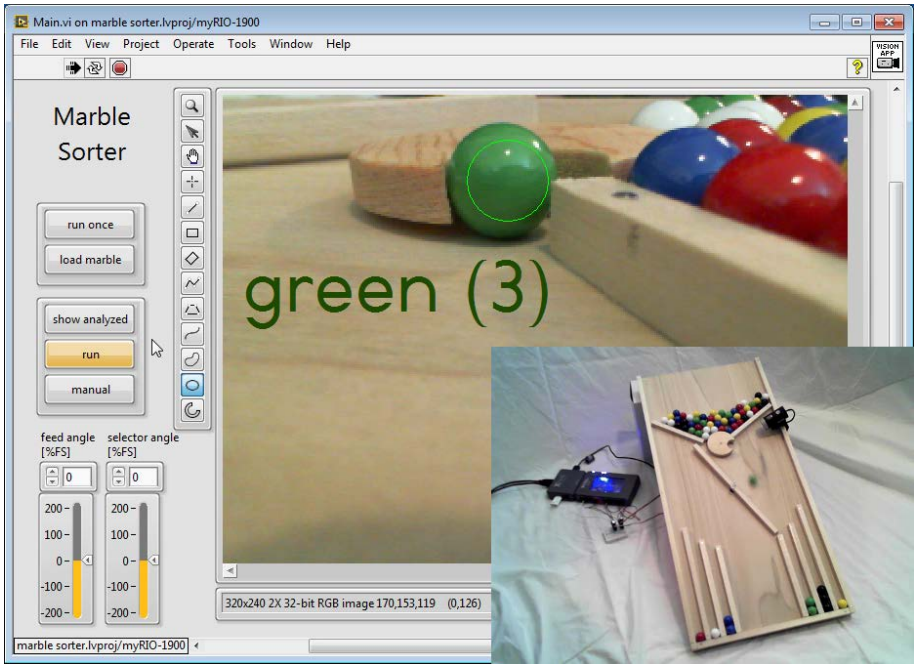

Figure 17.1: Front panel of the *Marble Sorter* application.

Functional Requirements: The *Marble Sorter* will:

1. Sort a hopper containing a mixture of marbles with up to six distinct colors into bins of the same color using this four-step sequence:
   (a) Rotate the feed disk under the hopper to load one marble into the feed disk slot,
   (b) Rotate the feed disk to present an unobstructed view of the marble to the webcam for color analysis,
   (c) Rotate the bin selector arm to the appropriate bin, and
   (d) Rotate the feed disk to drop the marble onto the bin selector arm which in turn guides the marble to its proper bin.
2. Load a single marble into the feed disk and present it to the webcam for color analysis when load marble is clicked,
3. Run the four-step sorting sequence one time when run once is clicked,
4. Continually run the four-step sorting process when run is pressed,
5. Display the webcam image at the maximum possible frame rate while the marble sorter is idle,
6. Continually display the detected marble color and associated numerical index as a nondestructive overlay, even when the marble sorter is idle,
7. Enable manual control of the feed and selector servomotor angles when manual is pressed,
8. Accept a user-defined ROI in the image display for color analysis; use a default ROI in the absence of a user-defined ROI,
9. Idle the marble sorter when the feed disk is empty, and
10. Display the analyzed image when show analyzed is pressed, otherwise display the webcam image.

Figure 17.1 on the previous page shows the front-panel of the *Marble Sorter* application viewing a green marble for color analysis; the inset photo shows the marble sorter dropping the green marble into its bin.

Required Resources:

1. NI LabVIEW and NI Vision Assistant
2. NI myRIO with power supply and USB cable

3. USB webcam
4. USB hub (Stratom X-HUB or equivalent)
5. USB flash drive
6. Marble sorter hardware:
   (a) Marbles, standard 16 mm diameter with uniform color, e.g., *Chinese Checker Marbles*[1]
   (b) Two servo motors (*GWS S03N STD*[2]); refer to Chapter 17 of the *NI myRIO Project Essentials Guide*[3] for a complete discussion of servo motors and sample LabVIEW code for this servo
   (c) Marble track; see Appendix E on page 143 for construction details

Deliverables:

1. Lab report or notebook formatted according to instructor's requirements; include front-panel screen shots to demonstrate that your application meets functional specifications
2. .zip archive file containing all project files:
   (a) Representative images
   (b) Vision Assistant script
   (c) LabVIEW project file set

17.1 Technical Approach

Marble Sorter Hardware

The marble sorter hardware suggested for this project uses two servomotors to operate the *feed disk* and the *bin selector*. The feed disk contains a circular cutout (the *feed slot*) with room for a single marble, and the bin selector arm rotates to align with a selected bin at the bottom of the sorter. Elevate the top of the sorter by about 6 inches to provide sufficient "gravity power" for the marbles to fall through toward the bins without spilling out of the marble sorter.

The servomotor provides a little more than 180° of travel, therefore the feed disk pictured in Figure 17.2 on the next page can be driven to any angle between the "load" and "drop" positions. The feed disk should always be rotated

[1] http://www.amazon.com/gp/product/B000J4C370
[2] http://gwsus.com/english/product/servo/standard.htm
[3] http://www.ni.com/myrio/project-guide

completely to the "load" position, even if the feed slot sometimes fills earlier. This ensures that any remaining marbles on the left side of the hopper will fall into the feed slot. In addition, rotating the feed disk back and forth through the hopper agitates the queued marbles to reduce the chance of jamming. The "analyze" position presents an unobstructed view of the marble to the webcam, and the "drop" position releases the marble onto the selector arm. The bin selector arm pictured in Figure 17.3 on page 127 rotates to one of the six color bins depending on the results of color analysis by the vision system and guides the dropped marble into the selected bin.

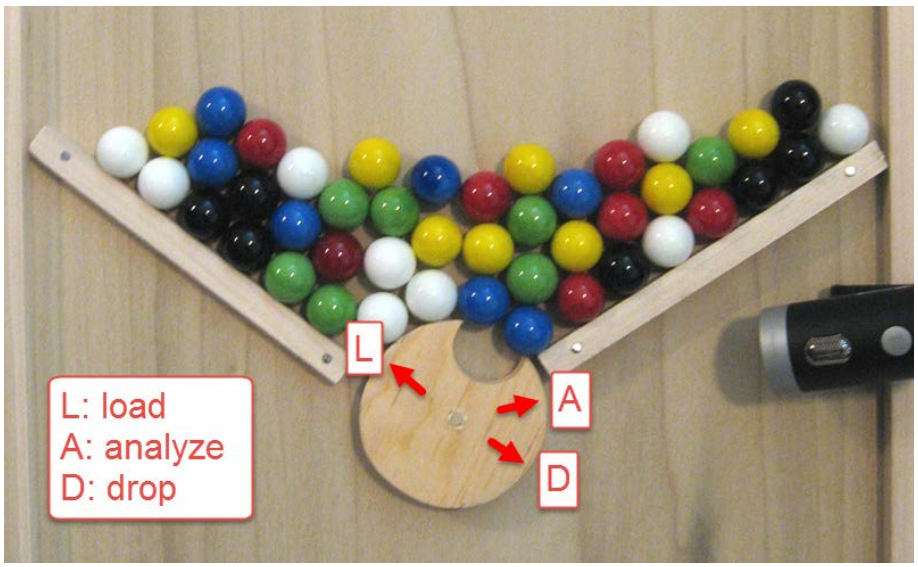

Figure 17.2: Marble hopper, feed disk, and webcam (right). The feed disk rotates to the "load" position to pick up one marble, then rotates back to the "analyze" position for color analysis by the vision system, and finally rotates to the "drop" position to send the marble to the bin selector.

NI Vision Assistant Script

The NI Vision Assistant *Color | Color Classification* step examines a region of interest (ROI) of an image and classifies the region as one of N colors defined by a classifier file (.clf suffix) created interactively with the NI Color Classification Training Interface. The training interface can work with multiple colors in a single image or with multiple images; this latter mode is appropriate for multiple webcam images of the feed disk rotated to the "analyze" position with

each image containing a marble of a different color along with an image of the empty feed disk slot.

The *Color Classification* step provides inputs for a user-defined ROI and color classifier file path when embedded in the NI Vision Assistant Express VI. These two inputs offer flexibility to redefine the marble's ROI should the webcam position change and to easily change the classifier with changes in lighting conditions.

Study the tutorial video *Classify Colors*[4] to learn how to create the color classifier file.

Servo Control

Chapter 17 of the *NI myRIO Project Essentials Guide*[5] describes how to set the angle of a servomotor with the NI myRIO PWM (pulse-width modulated) output; see *Servo Interfacing Theory*[6] for details. Use the block diagram code described in that chapter to convert a desired angle expressed in percent of full-scale rotation (-100 to $+100$) into the corresponding duty cycle of the PWM output; see *Servo Demo Walk-Through*[7] to learn more about the code which may be downloaded at *NI myRIO Project Essentials Guide .zip Files*[8].

The demo code uses the low-level PWM VIs Open and Set Duty Cycle and Frequency. See the NI myRIO low-level PWM subpalette to find the related VIs Set Duty Cycle and Close. You may wish to modify this demo code to add a second PWM channel as a convenient way to experiment with the two marble sorter servomotors and to obtain the needed servo angles for the feed disk and the bin selector arm. Spend some time manually controlling the marble sorter to develop a better feel for how to design the automation in the complete *Marble Sorter* application.

> **TIP**: *Refer to the NI myRIO connector diagrams in Appendix B on page 137.*

> **IMPORTANT**: *Servomotors generate a significant amount of electrical noise that may cause interference (cross talk) between the two PWM channels that control the servomotor angles. To remedy this problem, use a*

[4] http://youtu.be/MROZNOqDY_Y (4:56)
[5] http://www.ni.com/myrio/project-guide
[6] http://youtu.be/DOu5AvSDP2E (7:18)
[7] http://youtu.be/QXHe0DFbUdc (4:23)
[8] http://www.ni.com/academic/myrio/project-guide-vis.zip

100 μF electrolytic capacitor as a bypass (decoupling) capacitor between the NI myRIO 5-volt supply and ground. Be careful to observe proper polarity of the electrolytic capacitor!

Create the set servo angle VI to simplify the interface to your two servomotors and to speed development of your *Marble Sorter* application. The VI accepts the custom control servo info (consists of a PWM channel reference and a cluster array that contains the servomotor angle, delay time in milliseconds, and text label for each servomotor position) and a numerical index for the cluster array. For example, the feed disk servo info data highway lane would be initialized in the ***(qsm) initialize*** state with the PWM channel reference for the feed disk servomotor and a cluster of values for each of the three positions "load," "analyze," and "drop;" the delay times indicate the amount of time to wait for the servo to finish rotating to the indicated position (see the built-in LabVIEW VI Wait (ms)) and the text string labels define the name of each position. Running set servo angle with this servo information and index "1" would cause the feed disk servomotor to begin rotating to the "analyze" position and then wait for the designated number of milliseconds to allow the servomotor to complete its motion.

TIP: *Use the "Quick Drop" shortcut (Ctrl + space bar) to look up and place a LabVIEW element by name. See Appendix C on page 139 for more LabVIEW tips.*

In a similar fashion, the bin selector servo info data highway lane would be initialized with a PWM channel reference for the bin selector servomotor and the six angles, delay times, and text labels for the marble bins. Running set servo angle with this servo information and index "2" would cause the bin selector servomotor to rotate to the third bin and then wait for the designated number of milliseconds.

Refer to *Create "Coin Info" Cluster Array*[9] to learn the general procedure necessary to create the servo info custom control and to initialize its values. Because the PWM reference of the low-level PWM VIs is a type definition, follow the procedure illustrated beginning at 2:10 of *Step 5: Connect Script*[10] to insert this pre-defined control into your custom control.

See *Create "Flag Change" SubVI*[11] for a complete example illustrating the necessary techniques to create your own set servo angle VI. Be sure to set the execution mode to "Preallocated clone reentrant execution" as illustrated at time 6:40.

[9] `http://youtu.be/Yqv4u5FE8Vk` (5:12)
[10] `http://youtu.be/TWYBgLmCmnc` (3:53)
[11] `http://youtu.be/N1hUkWeZ8x0` (6:58)

LabVIEW Techniques

The following LabVIEW techniques are required for this project:

- *Get ROI from Image Display or Default*[12] – Get the ROI (region of interest) from the main image display with a "Property Node." Use a default ROI defined by the Vision Assistant script when no ROI is defined on the image display.
- *Overlay Text*[13] – Create a nondestructive text overlay with the "IMAQ Overlay Text" VI.
- *Select Overlay Text Font*[14] – Select the "NI Vision" user-defined text font and set its attributes such as font size and alignment.

17.2 Development Tips

Representative Images

- Implement the manual servomotor control section of your application first so that you can set especially the feed disk to arbitrary angles. Decide on the angle you will use for the "analyze" position.
- Set the webcam mode to a relatively low resolution for high frame rate. In this application the webcam is nothing more than a color sensor, therefore high resolution is not required.
- Set all of the webcam attributes to manual, especially "Focus" and "White Balance," to ensure consistent color sensing.
- Right-click on the image display and choose "Save Image," repeating as needed to acquire images of each marble color as well as the empty feed disk slot.

Vision Assistant Script

- The *Color Classification* step produces a string based on the class labels that you select. Use meaningful class names such as "red," "white," and "blue" for readability. However, you may also want to consider using numerical values for class names, e.g., "0," "1," "2," and so on because the LabVIEW Decimal String To Number function can directly translate these

[12] `http://youtu.be/p1sm8iVpVfo` (2:59)
[13] `http://youtu.be/SS72-mKX6fQ` (2:15)
[14] `http://youtu.be/lEKJcR553zA` (2:48)

class names into a numerical color index for the purpose of looking up bin selector servomotor angles from an array. Get the best of both worlds by using class names of the form "0: red," "1: white," and "2: blue" because the VI can pick out a decimal number from the surrounding text.

- When packaged into the Vision Assistant Express VI the color classifier produces an array of strings as output because the classifier can work with compound ROIs. Use the Index Array function to pick out a single classification label.

Optional Extra Features:

You may wish to add one or more of these extra features:

- Display the sensed color code index as a binary pattern on the NI myRIO on-board LEDs; see Number to Boolean Array.
- Add the LCD display to show the sensed color code; see Chapter 8 on page 57 for details on the LCD.
- Maintain the current position of the bin selector servomotor so that the wait time can be eliminated when the next marble color is the same as the previous marble color. In other words, when several marbles of the same color pass through the feed disk it is unnecessary to wait for bin selector arm motion because it is already in the correct position.

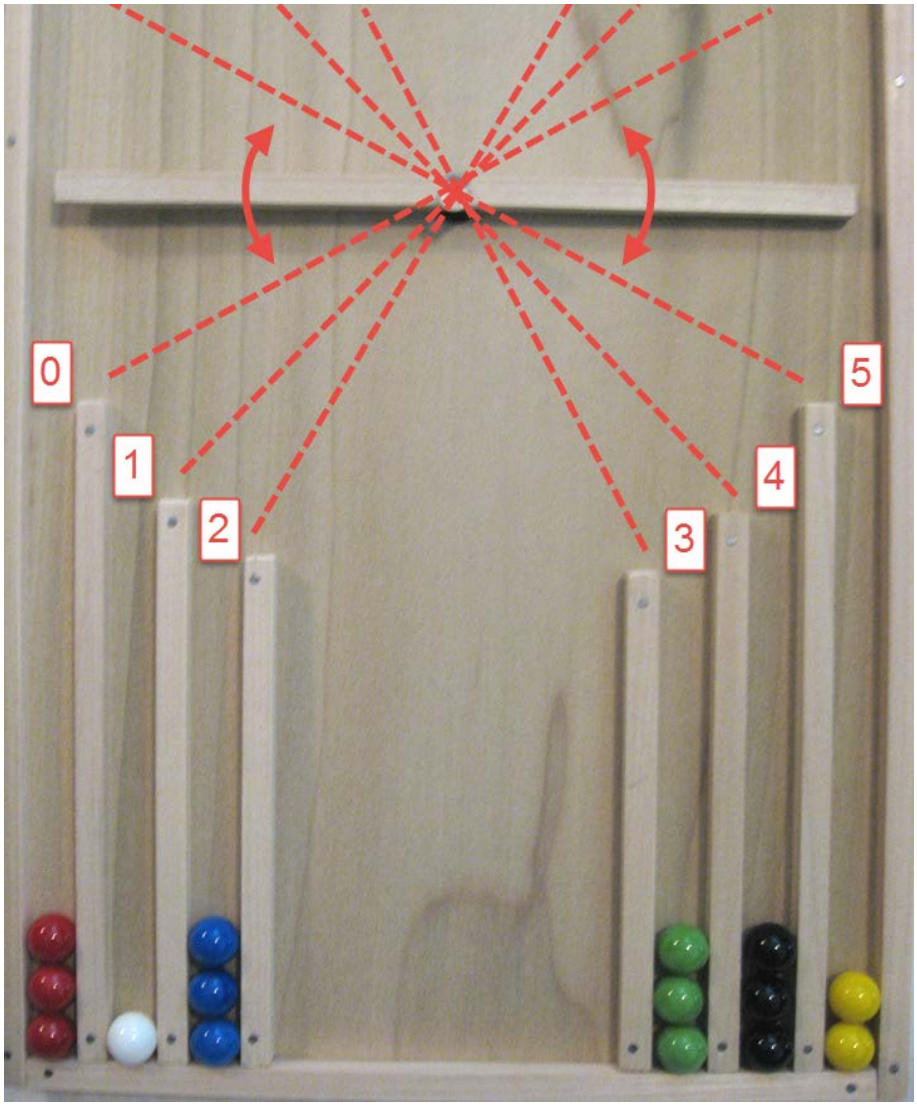

Figure 17.3: Bin selector arm located under the feed disk. The bin selector rotates to one of the six color bins depending on the color analysis result of the vision script.

Part IV

Appendices

A Recommended Equipment

A.1 USB Webcam

- *Microsoft LifeCam Studio Webcam*[1] pictured in Figure A.2 on the following page; search the major resellers such as Amazon to find the best price on this product
- Desirable features:
  - Wide range of resolutions from 176×144 to 1920×1080
  - Electronically-controlled optical focus (critical for machine vision projects!)
  - Standard threaded hole (1/4"-20) for mounting to copy stand or tripod
- Practical considerations:
  - In my experience 1280×720 is the highest practical resolution and I often try to get by with 800×600 or 640×480. The highest resolution of 1920×1080 has a maximum frame rate of three to five frames per second simply to receive and display the source video, and subsequent processing can bring the frame rate down considerably due to the sheer volume of required myRIO CPU time.
  - The standard threaded hole is very handy as a mount to the copy stand or tripod, however, precise alignment of the camera is very difficult and the camera tends to bounce and vibrate because the mounting bracket acts like spring. Use a block of material and duct tape as suggested in Figure A.1 on the next page to stabilize the camera against the threaded hole.

[1] `http://www.microsoft.com/hardware/en-us/p/lifecam-studio`

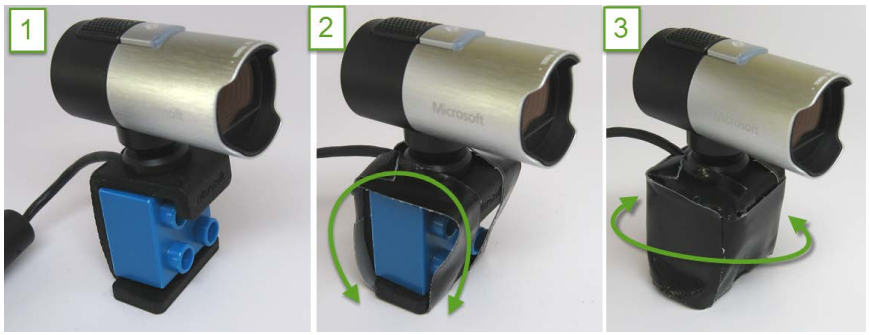

Figure A.1: Webcam stabilizer: (1) Insert a plastic block of material, (2) wrap with two narrow strips of duct tape, and (3) wrap with a single wide piece of duct tape.

A.2 Tabletop Camera Copy Stand

Figure A.2: Microsoft LifeCam Studio webcam mounted to the RPS RS-CS305 tabletop camera copy stand.

- *RPS RS-CS305 Copy Stand*[2] pictured in Figure A.2 on the facing page
  - One 1/4"-20 threaded camera mount with plastic tightening wheel
  - Telescoping column (6" to 12") for adjustable camera height
  - Two-position camera mounting bracket to extend the range of baseboard-to-camera distance
  - 9"×12" laminated baseboard
- Desirable features:
  - Compact size
  - Reasonably priced
  - Provides for a wide range of camera-to-object distances
- Practical considerations:
  - Use a paper or fabric background at all times. The laminated baseboard surface scratches easily and would soon become filled with "optical noise." Also, the slightly reflective laminate surface often causes unwanted artifacts in the acquired image.

A.3 Grid Calibration Targets

- *1×1 cm Lined Grid*[3]; see *Incompetech's Grid Paper*[4] for additional target patterns.
- 1×1 cm Dot Grid: look for the "CalibrationGrid.pdf" file in the "Vision Documentation" subfolder of your LabVIEW installation folder.
- Desirable features:
  - Use the lined grid to align the webcam and minimize tangential distortion, i.e., when the optical axis of the camera is not normal to the object plane
  - Use the dot grid to perform grid calibration (corrects lens distortion and tangential distortion) with NI Vision Assistant
- Practical considerations:

[2] http://www.amazon.com/Copy-Stand-Tabletop-Desktop-9x12/dp/B003OAF2BA

[3] http://incompetech.com/graphpaper/custom/centimeter-black.pdf

[4] http://incompetech.com/graphpaper

  - Be certain to print the grids with 100% scaling; measure the printed grids to confirm accurate printing

A.4 Black Nonreflective Paper Background

- *Black Suede Bulk Paper*[5]
  - 8.5"×11"
- Desirable features:
  - This black velvet-on-paper provides an excellent nonreflective background surface with negligible pattern.
- Practical considerations:
  - I discovered this product at our local Jo-Ann Fabric and Crafts store.
  - Cover when not in use as the velvet will rather quickly pick up dust. Use a lint roller, e.g., *3M Lint Roller*[6] to clean the surface just before you acquire images.
  - Black felt works as a reasonable replacement, although it has a bit more visible pattern

A.5 USB Hub

- *Stratom X-HUB*[7], pictured in Figure A.3 on the next page
  - Three USB ports
  - One Ethernet port
- Desirable features:
  - Designed for direct connection to NI myRIO: X-HUB replicates the existing USB connectors and power supply jack
  - Extends the NI myRIO USB port count to three (instead of one) and adds an Ethernet port

[5] http://www.hotp.com/products/10358

[6] http://www.amazon.com/3M-836R-OS-Lint-Roller/dp/B00006IA8Q/ref=sr_1_11

[7] http://sine.ni.com/nips/cds/view/p/lang/en/nid/213322

Figure A.3: Stratom X-HUB 3-port USB hub and Ethernet port expander.

- Practical considerations:
  - Required for any machine vision project that needs a USB flash drive

A.6 Stereo Camera Bracket

- *Desmond D3D-3 Mini Stereo Camera Bracket*[8], pictured in Figure A.4 on the following page
  - Two 1/4"-20 threaded camera mounts with plastic tightening wheels
  - One 1/4"-20 and one 3/8"-16 mounting threads
  - 8-inch length
- Desirable features:
  - Connects directly to the RPS RS-CS305 copy stand
  - Connects directly to the Microsoft LifeCam Studio webcams

[8] http://www.amazon.com/gp/product/B00BCSTGUE/ref=ox_sc_act_title_1

Figure A.4: Desmond D3D-3 mini stereo camera bracket.

  - Provides a range of stereo webcam baselines

- Practical considerations:
  - Required for stereo vision machine vision projects

B MXP and MSP Connector Diagrams

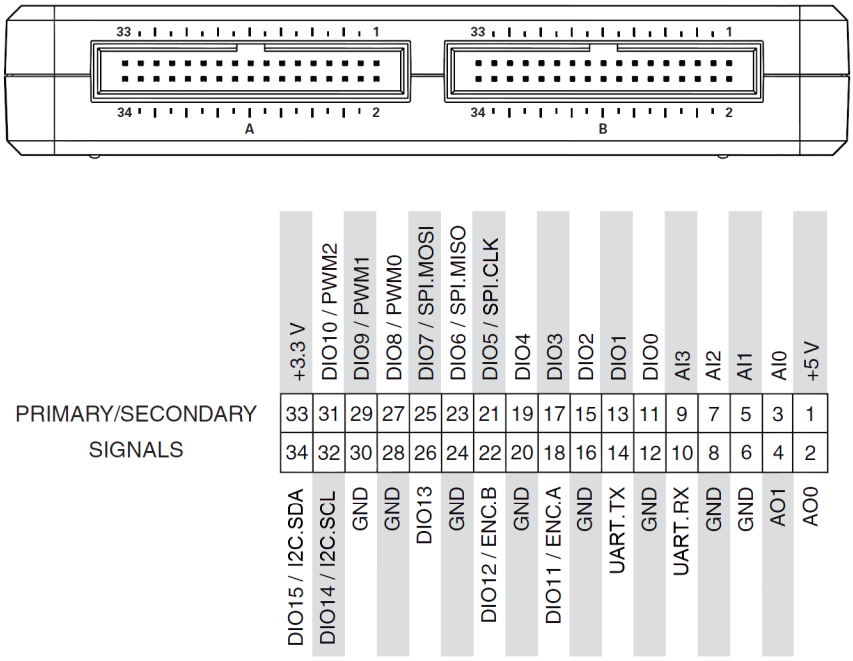

Figure B.1: MXP (**m**yRIO e**X**pansion **P**ort) connector diagram.

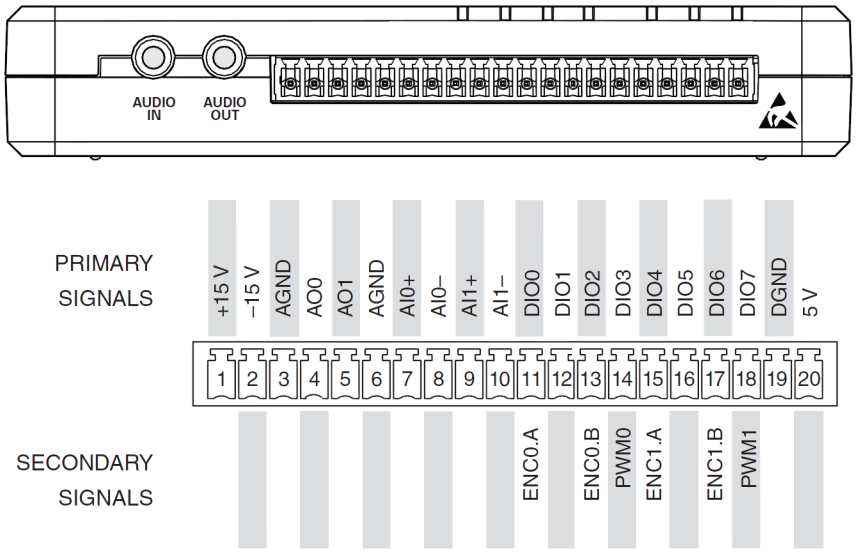

Figure B.2: MSP (**m**ini**S**ystem **P**ort) connector diagram.

C LabVIEW Quick Tips

LabVIEW offers a huge number of keyboard shortcuts to help you to program more efficiently. This appendix lists some of the most commonly-used shortcuts. You may also wish to print out this *LabVIEW Quick Reference Card (2010)*[1].

C.1 Find VIs and learn how they work

- Quick Drop: **Ctrl+Space** in block diagram, start typing the name of the VI
- Context Help: **Ctrl+H** to toggle on and off, hover cursor over any object or wire
- Detailed Help: Right-click on the object and choose Help

C.2 Organization and tidying up

- Align objects: **Ctrl+Shift+A** to invoke the most-recent alignment type; click "Align Objects" in the top center of the LabVIEW window to select the desired alignment method
- Distribute objects: **Ctrl+D** to apply equal spacing between the selected objects; click "Distribute Objects" in the top center of the LabVIEW window for additional spacing methods
- Copy with alignment: **Ctrl+Shift+[left mouse drag]** on object or selected region; drag either horizontally or vertically
- Move selected object or objects in small steps: arrow keys
- Move selected object or objects in larger jumps: **Shift+[arrow keys]**

[1] `http://digital.ni.com/manuals.nsf/websearch/5825145A9C56A01A8625770E00747555`

- Open up more space between objects: **Ctrl+[left mouse drag]** in white space between objects; drag horizontally to open up horizontal space, same idea for vertical space, and drag diagonally to open up horizontal and vertical space at the same time
- Clean up wires: Right-click on the wire, choose "Clean Up Wire"
- Delete broken wires: **Ctrl+B** to delete all broken wires or pre-selected wires
- Clean up the diagram: **Ctrl+U** or click the broom icon ("Clean Up Diagram" in the top center of the LabVIEW block diagram editor) to automatically align the entire diagram or a pre-selected region
- Dimensioned controls and indicators: Include units in the label, e.g., "frequency [Hz]"

C.3 Power-user efficiency

- Undo last action: **Ctrl+Z**; repeat as needed to undo earlier actions
- Create a new control, indicator or constant: Right-click on a terminal, select "Create," and then choose the desired object
- Change existing control to constant, etc.: Right-click, choose "Change To Constant" (options change depending on whether you select a control, constant, or indicator)
- Replace object with another from the same palette: Right-click, choose "Replace," select the palette name, and then choose the new object
- Pan and scroll the diagram: **Ctrl+Shift+[left mouse drag]**
- Pin frequently-used palettes: Click the thumbtack symbol at the upper-left corner of any palette
- Set default values for all front-panel controls: Select Edit|Make Current Values Default; you can also set default values only for selected controls, too
- Printing: File|Print and choose "Icon, description, panel and diagram" for basic printing or "VI documentation" for full control over the print appearance; click the "Preview" button at any time to see how the print job will look

D Build a Stand-Alone Application

During development you normally connect NI myRIO to your computer with a USB cable. Once development is complete you can easily deploy your project as a stand-alone application stored on the myRIO solid-state hard drive that starts automatically when you power up the myRIO; no USB cable required! Study the tutorial *Deploy a Stand-Alone Application*[1] to learn the step-by-step procedure to create a build for the real-time (RT) target, deploy the build as the startup application, and how to disable the startup application, if necessary.

[1] `http://youtu.be/JXoJECRS-eo` (8:29)

E Marble Sorter Construction Details

To be continued...

F Video Tutorial Links

F.1 Image Acquisition and Calibration

- *Camera Setup Principles*[1] – From the minimum feature size determine the necessary camera-to-object distance to obtain the desired spatial resolution and field-of-view (FOV).
- *Set Camera Defaults in NI-MAX*[2] – Use NI-MAX to set the USB webcam video mode (resolution and frame rate), select color (RGB) or grayscale acquisition, set the camera attributes (brightness, contrast, focus, etc.), and save as the default for all NI vision-related programs.
- *Acquire Image in NI Vision Assistant*[3] – Acquire a USB webcam image within NI Vision Assistant; measure a distance on the image.

F.2 Stereo Vision

- *"Compute Depth Image" Example VI*[4] – Demonstrate the "Compute Depth Image" example VI included with the NI Vision installation that illustrates how to measure depth with two stereo cameras.
- *Configure Stereo Cameras*[5] – Configure the "Vision Acquisition" Express VIs of the "Calibrate Stereo Vision" example VI (included with the NI Vision installation) to use your specific pair of webcams.

---

[1] `http://youtu.be/ul5sme31cMo` (9:16)
[2] `http://youtu.be/oTWHhMENsCg` (3:51)
[3] `http://youtu.be/wTJr8beDtvQ` (3:58)
[4] `http://youtu.be/TEihyEGrsqQ` (3:35)
[5] `http://youtu.be/iGqlBrg3DjM` (2:47)

- *Calibrate Stereo Cameras*[6] – Calibrate a pair of stereo cameras for use with the "Calibrate Stereo Vision" example VI included with the NI Vision installation.
- *Measure Depth*[7] – Measure depth using a pair of stereo cameras and the "Calibrate Stereo Vision" example VI included with the NI Vision installation.

F.3 Coin Caliper I

- *Step 1: Obtain Representative Images*[8] – Coin Caliper I Demo Step 1/7: Optimize the camera contrast, acquire representative images of coins and of a ruler to be used later for calibration.
- *Step 2: Calibrate the Camera*[9] – Coin Caliper I Demo Step 2/7: Create a "Point-Distance" calibration step in NI Vision Assistant and measure length in real-world units.
- *Step 3: Develop Vision Script*[10] – Coin Caliper I Demo Step 3/7: Develop the Vision Assistant script to convert the color webcam image to grayscale and then measure the coin diameter with the "Max Clamp" step; test the vision script with the representative images collected earlier.
- *Step 4: Create LabVIEW Project*[11] – Coin Caliper I Demo Step 4/7: Create a new LabVIEW project for execution on the NI myRIO real-time target, and then prepare the block diagram.
- *Step 5: Configure "Vision Acquisition" Express VI*[12] – Coin Caliper I Demo Step 5/7: Place the "Vision Acquisition" Express VI and ensure that the camera settings match those used to acquire the representative images.
- *Step 6: Configure "Vision Assistant" Express VI*[13] – Coin Caliper I Demo Step 6/7: Place the "Vision Assistant" Express VI and insert the vision script developed earlier.

[6] `http://youtu.be/bufXESBlzTY` (2:49)
[7] `http://youtu.be/UXw31LOSiHk` (3:16)
[8] `http://youtu.be/DMH5Z-J0mcA` (4:21)
[9] `http://youtu.be/2ycFJJcKbMs` (3:04)
[10] `http://youtu.be/AO5Kh-VN5sA` (2:59)
[11] `http://youtu.be/zZ1oLiItM3g` (1:31)
[12] `http://youtu.be/DmUHugQ6qJM` (3:34)
[13] `http://youtu.be/_il9DoPwkpM` (4:37)

- *Step 7: Add I/O Devices*[14] – Coin Caliper I Demo Step 7/7: Use the myRIO onboard LEDs as external indicators to indicate when the coin diameter exceeds 23 mm.

F.4 "Machine Vision App" (MVA) Template

LabVIEW Project Files Overview

- *MVA Project Files and Folders*[15] – Open the "machine vision app" folder, review the files and folders, open the .lvproj file, and view the organized files.
- *MVA Main.vi Quick Tour*[16] – Quick-paced tour of the "Main" VI: while-loop, case-structure, state queue, front-panel controls, NI myRIO ports, and data highway.
- *MVA States*[17] – Review the standard states (qsm, myRIO, image), the example application-specific state, and the expansion states.
- *Run the MVA Project*[18] – Open the "machine vision app" folder, open the .lvproj file, open Main.vi, run the VI, demo the buttons, and stop. NOTE: The webcam must already be configured!
- *Run the MVA Project (New Camera)*[19] – Run the first time with a new camera: Use NI-MAX to generate the configuration files, and then configure Vision Acquisition Express VI.
- *"Initialize Queue" VI*[20] – Walk through the "initialize_queue" VI.
- *"Enqueue States" VI*[21] – Walk through the "enqueue_states" VI.
- *"Dequeue State" VI*[22] – Walk through the "dequeue_state" VI.
- *Change Camera Mode*[23] – Change the camera mode from color to grayscale and also change the resolution; demonstrate where to remove the "extract luminance" step in the Vision Assistant Express VI.

[14] `http://youtu.be/0RAIwdW5byk` (3:06)
[15] `http://youtu.be/AekhpmU9s3Y` (2:04)
[16] `http://youtu.be/sgGgU2dkocI` (4:52)
[17] `http://youtu.be/IKFeOF4Zh6Y` (2:18)
[18] `http://youtu.be/MITs9SQz-Pg` (1:38)
[19] `http://youtu.be/Brfnek3mgoA` (4:11)
[20] `http://youtu.be/p97xHbXmMgg` (1:20)
[21] `http://youtu.be/AFg-_tv9MVU` (2:16)
[22] `http://youtu.be/-uIB_5KlstQ` (2:30)
[23] `http://youtu.be/vUTFTDHy4Mc` (2:49)

Standard States

- *(qsm) initialize: Image Buffer*[24] – (qsm) initialize activity: Create an image buffer with "IMAQ Create" and then merge the reference into the data highway.
- *(qsm) initialize: Front-Panel*[25] – (qsm) initialize activity: Initialize a front-panel control or indicator with an "Invoke Node."
- *(qsm) release*[26] – (qsm) release: This state releases the front-panel action buttons at the end of a task.
- *(qsm) cleanup*[27] – (qsm) cleanup: Set the "Stop webcam flag" and close an open file, for example.
- *(qsm) shutdown*[28] – (qsm) shutdown: Release the state queue, free all image buffer memory, and stop the while-loop.
- *(qsm) schedule*[29] – (qsm) schedule: Select a task to apply to the "enqueue_states" VI and create a new task.
- *(myRIO) initialize*[30] – (myRIO) initialize: Example activities such as sending I2C data and setting up a PWM channel.
- *(myRIO) cleanup*[31] – (myRIO) cleanup: Example activities such as send digital and analog outputs to inactive states, close PWM channel, send I2C shutdown string.
- *(myRIO) update ports*[32] – (myRIO) update ports: Example activities such as reading and writing onboard devices, digital I/O, analog I/O, serial buses, PWM, etc.
- *(image) get*[33] – (image) get: Walk through the details of this state, including the Vision Acquisition Express VI and "IMAQ Symmetry" VI.
- *(image) get: Vision Acquisition*[34] – (image) get: Use NI-MAX to create and update the camera configuration files stored on the myRIO target;

[24] `http://youtu.be/suCkeB_afOc` (2:02)
[25] `http://youtu.be/eBtBy__PNwU` (1:49)
[26] `http://youtu.be/NN4qyYiG7CM` (2:13)
[27] `http://youtu.be/yA9TaFt6lYQ` (1:22)
[28] `http://youtu.be/YgLynyjlPo8` (1:30)
[29] `http://youtu.be/uS2QlWo3VM4` (4:27)
[30] `http://youtu.be/KtGDT-jTtmg` (1:40)
[31] `http://youtu.be/fi1GUcoloj0` (1:12)
[32] `http://youtu.be/3gNSpF4RnXA` (1:49)
[33] `http://youtu.be/guqZisxPUGw` (2:23)
[34] `http://youtu.be/jrhG9sOwGIQ` (3:13)

illustrate how to change the camera mode and attributes in NI-MAX and then retrieve these values with Vision Acquisition Express VI.

- *(image) analyze*[35] – (image) analyze: Illustrate how to load a Vision Assistant script into the Express VI.
- *(image) overlay*[36] – (image) overlay: Show VIs to create overlay text and graphics and discuss details of the VIs in this state (time stamp, overlay text, format into string, merge errors).
- *(image) show*[37] – (image) show: Explain how to display one of the available image buffers on the main image display.

Edit the Data Highway

- *Change Data Lane Name*[38] – Change a data highway lane name: edit the type def, and then observe the change in the bundle/unbundle functions.
- *Add Data Lane*[39] – Create a new data highway lane: edit the type def, bundle and merge, and access in another state.
- *Remove Data Lane*[40] – Remove an unused data highway lane: edit the type def, delete the control, remove the element in the cluster bundle.
- *Add Custom Data Lane*[41] – Add a custom lane: create a control from the Vision Assistant Express VI indicator output, convert to type def, save type def as a control, add the control to the data highway, merge the Vision Assistant Express VI output onto the data highway, and then retrieve the value in another state.

Edit the States

- *Change State Name*[42] – Change a state name: edit the subdiagram label, open the state type def, edit the enumerated datatype control, observe the changed name in the case-structure subdiagram.

[35] `http://youtu.be/mKuwsgGs7sY` (2:23)
[36] `http://youtu.be/Vmony2qtnH8` (3:12)
[37] `http://youtu.be/LrjuylUE_4Q` (1:31)
[38] `http://youtu.be/q2iusxCE3Yo` (0:54)
[39] `http://youtu.be/7UFQNZJDYYU` (1:27)
[40] `http://youtu.be/Kt6aeX3d4Yg` (0:47)
[41] `http://youtu.be/GkqjZCj3xck` (2:58)
[42] `http://youtu.be/pj_aboGCvjI` (1:09)

- *Rearrange States*[43] – Reposition a state in both the state type def and in the case structure's subdiagram ordering.
- *Add State at End*[44] – Add a state at the end: open the state type def, edit the enumerated control, show two ways to create a new case-structure subdiagram.
- *Remove State*[45] – Remove an unused state in a task, in the case structure, and in the state type def.
- *Remove State with Review/Accept Changes*[46] – Remove an unused state; "Review and Accept Changes" message.

Tasks

- *Add Task and Action Buttons*[47] – Add a new task and action button: Add a button, extend the selector tree, add a new task (array constant).

Step-By-Step Demo (Coin Caliper II)

- *Step 1: Rename and Open Project*[48] – Coin Caliper II demo Step 1/15: Download the "Machine Vision App" (MVA) template, rename the folder, rename the .lvproj file, open the project, rename the while-loop structure, and rename the app.
- *Step 2: Interpret Specifications*[49] – Coin Caliper II demo Step 2/15: Interpret the functional specifications.
- *Step 3: Develop Vision Script*[50] – Coin Caliper II demo Step 3/15: Develop the vision script in the stand-alone version of Vision Assistant to measure and report the coin diameter in pixels and in real-world units.
- *Step 4: Import Vision Script*[51] – Coin Caliper II demo Step 4/15: Import the vision script into the LabVIEW Vision Assistant Express VI; select controls and indicators.

[43] `http://youtu.be/z6wO6RQy3QY` (1:25)
[44] `http://youtu.be/jJlEtD-2zgU` (1:49)
[45] `http://youtu.be/F8kjxt0mHTI` (1:23)
[46] `http://youtu.be/G_k7aLgglgE` (2:03)
[47] `http://youtu.be/Y_u6FzX2C8M` (2:26)
[48] `http://youtu.be/YzbmvF5ZLUA` (1:22)
[49] `http://youtu.be/sbP45Merxxs` (4:45)
[50] `http://youtu.be/LR33-W4Thww` (5:12)
[51] `http://youtu.be/sIj-EPxV8Tg` (2:24)

- *Step 5: Connect Script*[52] – Coin Caliper II demo Step 5/15: Connect the Vision Assistant Express VI controls and indicators to the data highway.
- *Step 6: Obtain ROI*[53] – Coin Caliper II demo Step 6/15: Use a "Property Node" to obtain the user-selected ROI from the main image display.
- *Step 7: Create Task Buttons*[54] – Coin Caliper II demo Step 7/15: Create an action button for the "calibrate" state.
- *Step 8: Create Increment State*[55] – Coin Caliper II demo Step 8/15: Create the "increment file number" state and add "file number" to the data highway.
- *Step 9: Create Calibrate State*[56] – Coin Caliper II demo Step 9/15: Create the "calibrate" state, add "known diameter" to the data highway, and calculate the scale factor as known diameter [mm] divided by measured diameter in pixels.
- *Step 10: Edit Overlay State*[57] – Coin Caliper II demo Step 10/15: Edit the "(image) overlay" state to add the measured coin diameter to the nondestructive overlay.
- *Step 11: Edit Save Image State*[58] – Coin Caliper II demo Step 11/15: Edit the "save image" state to obtain the base image name from a front-panel control and the "file number" from the data highway.
- *Step 12: Edit myRIO Ports State*[59] – Coin Caliper II demo Step 12/15: Edit the "(myRIO) update ports" state to operate the myRIO LED indicators.
- *Step 13: Edit Scheduler State*[60] – Coin Caliper II demo Step 13/15: Edit the "(qsm) schedule" state to create the necessary tasks.
- *Step 14: Set Up Camera*[61] – Coin Caliper II demo Step 14/15: Set up the camera mode (resolution) and attributes.
- *Step 15: Run and Debug*[62] – Coin Caliper II demo Step 15/15: Run and debug the application until it works properly.

[52] `http://youtu.be/TWYBgLmCmnc` (3:53)
[53] `http://youtu.be/LzvU9eOcXO8` (1:32)
[54] `http://youtu.be/8WKzPgNUbQ4` (1:07)
[55] `http://youtu.be/5yp9NaZA7SI` (2:45)
[56] `http://youtu.be/pc2DZfITPrM` (2:43)
[57] `http://youtu.be/DgJ-ere6_5k` (2:19)
[58] `http://youtu.be/JwuVK83GLJo` (4:40)
[59] `http://youtu.be/7LFFkM7CZBg` (2:42)
[60] `http://youtu.be/jLb0lOpKnx0` (2:17)
[61] `http://youtu.be/jEXkhzawvNE` (2:54)
[62] `http://youtu.be/3gbpGOIED5U` (8:19)

F.5 NI Vision Assistant

Batch Processing

- *Batch Process a Folder of Images*[63] – Apply a Vision Assistant script in batch mode to process a set of images, e.g., flip the image geometry of a set of images.

Binary Images

- *Remove Small Objects*[64] – Remove small objects from a binary image with advanced morphology.
- *Fill Holes*[65] – Fill holes in objects in a binary image.
- *Binary Circle Detector*[66] – Detect circles in a binary image.
- *Report Area of Particles*[67] – Use the "Particle Analysis" step to report the number of particles found in the image and the area of each particle.
- *Remove Small-Area Particles*[68] – Use the "Particle Filter" to remove small noise particles based on the particle area measure.

Color Images

- *Classify Colors*[69] – Classify the color of a region of interest (ROI) according to classes defined by the color classifier (.clf) file.
- *Extract Luminance Plane*[70] – Extract the luminance plane from a color image as the grayscale version of the color image.
- *Check for Matching Color*[71] – Check a region of interest (ROI), either single or compound, to determine whether or not the ROI color matches an expected color.

[63] `http://youtu.be/xee96nDrUpw` (2:22)
[64] `http://youtu.be/Xfo9wK_4B7Q` (1:09)
[65] `http://youtu.be/GjwostpEeTo` (1:35)
[66] `http://youtu.be/9fo76IeeUMM` (1:47)
[67] `http://youtu.be/N8T3HDdlN-E` (1:44)
[68] `http://youtu.be/2v1sgSew8cs` (2:13)
[69] `http://youtu.be/MROZNOqDY_Y` (4:56)
[70] `http://youtu.be/1mmzsd46_Gs` (1:02)
[71] `http://youtu.be/VmO5fBoZMyE` (3:23)

Grayscale Images

- *Find Energy Center (Centroid)*[72] – Find the X-Y coordinates of the energy center (centroid) of an image.
- *Suppress Highlights*[73] – Use the "Grayscale | Operators" with "Min" option to suppress the highlights (hotspots) in a grayscale image. Pixel intensitiies that exceed a threshold are clipped to that threshold value.
- *Threshold Grayscale to Binary*[74] – Threshold a grayscale image to binary using manual or automatic threshold selection.
- *Dual-Threshold Grayscale to Binary*[75] – Threshold a grayscale image to binary using manual mode with dual (low and high) thresholds.

Identification

- *EAN-13 (UPC-A) Barcode Reader*[76] – Read EAN-13 (UPC-A) barcodes.
- *Code-39 Barcode Reader*[77] – Read Code-39 barcodes; demonstrate how to acquire images within the stand-alone version of Vision Assistant.
- *OCR (Optical Character Recognition) Training*[78] – Train the OCR reader; demonstrate how to acquire images within the stand-alone version of Vision Assistant.
- *QR Code Reader*[79] – Read QR codes (2-D barcodes); demonstrate how to acquire images within the stand-alone version of Vision Assistant.

Calibration, Real-World Units, and Coordinate Systems

- *Two-Point Calibration from a Ruler*[80] – Set up a 2-point calibration from an image of a ruler to convert measurements in pixels into real-world units such as millimeters.

[72] `http://youtu.be/XkYfR141x0k` (0:57)
[73] `http://youtu.be/XJaMjKxjLoo` (1:45)
[74] `http://youtu.be/7fJBS6_neQY` (3:08)
[75] `http://youtu.be/0TFh4zB6e3Y` (1:56)
[76] `http://youtu.be/hLSla3C-6V4` (1:29)
[77] `http://youtu.be/iT8FAzkSksE` (2:29)
[78] `http://youtu.be/KxRSL3jnIAk` (5:04)
[79] `http://youtu.be/8LTx9N0R8_g` (2:47)
[80] `http://youtu.be/kx45UGkbbbo` (2:28)

- *Grid Calibration*[81] – Set up a grid-based calibration to correct camera lens and tangential distortion: print the grid target, convert the image to grayscale, and evaluate the performance of the available distortion models.
- *Correct a Grid-Calibrated Image*[82] – Apply a geometric correction to an image based on a previous grid-calibration step.
- *Set Coordinate System*[83] – Set the coordinate system relative to a standard feature of an object, e.g., a corner defined by two edges.

Machine Vision

- *Find the Intersection of Two Lines*[84] – Use the "Caliper" to find the coordinates of the intersection of two straight lines found in previous steps.
- *Measure the Distance Between Two Points*[85] – Use the "Caliper" to measure the distance between two points.
- *Measure a Perpendicular Projection Distance*[86] – Use the "Caliper" to measure the perpendicular projection distance between a baseline defined by two points and a third point.
- *Find Midpoint Between Two Points*[87] – Use the "Caliper" to find the midpoint between two points.
- *Measure the Angle Formed By Two Lines*[88] – Use the "Caliper" to measure the angle between two lines defined by four points.
- *Multiple Caliper Measurements*[89] – Use the "Caliper" to make multiple measurements in a single step.
- *"Gauging Demo" Connections*[90] – Connect the "Gauging Demo" Vision Assistant script to the Express VI in LabVIEW.

[81] `http://youtu.be/eWqq65ZZ0NQ` (6:47)
[82] `http://youtu.be/mJFsqqd37WY` (1:34)
[83] `http://youtu.be/PDcWiz2_mn0` (2:02)
[84] `http://youtu.be/uOEhkFDuhVQ` (2:04)
[85] `http://youtu.be/4AnBurRhXe8` (1:46)
[86] `http://youtu.be/o6tNTR8Znhw` (2:22)
[87] `http://youtu.be/sTdAFjSBUXI` (1:55)
[88] `http://youtu.be/r57NBz8F69U` (2:06)
[89] `http://youtu.be/ZbGumGRaRqU` (3:13)
[90] `http://youtu.be/auw8OBj_7m4` (6:54)

- *Measure Width with "Clamp (Rake)"*[91] – Measure the height or width of an object with the "Clamp (Rake)" step.
- *Measure Width with "Clamp (Rake)" (Rotated ROI)*[92] – Measure the height or width of an object with the "Clamp (Rake)" step that uses a rotated ROI referred to a local coordinate system.
- *Find Edge Locations*[93] – Use the "Edge Detector" step to determine the locations of edge features along a straight line.
- *Find Straight Edge*[94] – Find the straight edge of a part that may be translated or rotated.
- *Match Perimeter Contour*[95] – Look for a matching object based on similar perimeter contours.
- *Find Reference Feature (Geometric Matching)*[96] – Find a reference feature with "Geometric Matching" to define a local coordinate system.
- *Golden Template Comparison*[97] – Set up the "Golden Template Comparison" step.
- *Detect and Measure a Circular Shape*[98] – Detect a circular shape and measure its radius and center coordinates.

F.6 IMAQ Overlay VIs

Text

- *Overlay Text*[99] – Create a nondestructive text overlay with the "IMAQ Overlay Text" VI.
- *Overlay Text at X-Y Coordinates*[100] – Programmatically place overlay text at a specific X-Y location.

[91] `http://youtu.be/fPWQCnB7B4s` (2:18)
[92] `http://youtu.be/bJawZuGBxsU` (2:36)
[93] `http://youtu.be/6bcO-omuOi0` (3:21)
[94] `http://youtu.be/eJ3KVGSlnuY` (2:25)
[95] `http://youtu.be/dLTFKucEM_A` (7:08)
[96] `http://youtu.be/o84e5EkLAs4` (5:40)
[97] `http://youtu.be/Ux7mQGaEmkI` (4:42)
[98] `http://youtu.be/nNhkbTFt3cU` (2:06)
[99] `http://youtu.be/SS72-mKX6fQ` (2:15)
[100] `http://youtu.be/O5IIrPP_smA` (1:41)

- *Adjust Overlay Text Alignment*[101] – Adjust the horizontal and vertical alignment of an overlay text label.
- *Select Overlay Text Font*[102] – Select the "NI Vision" user-defined text font and set its attributes such as font size and alignment.

Graphics

- *Overlay Line*[103] – Create a nondestructive line overlay with the "IMAQ Overlay Line" VI.

Instrument Readers

- *Read a DMM Display*[104] – Read the display of a digital multimeter (DMM) with the "IMAQ Read LCD" VI.

F.7 LabVIEW Techniques

- *Concatenate Scalar to an Array*[105] – Concatenate a scalar to an array.
- *Math with "Position" Clusters*[106] – Perform basic mathematical operations with the "Position" cluster used by NI Vision tools.
- *Get Date and Time Strings*[107] – Get the system date and time as strings using the "Get Date/Time in Seconds" and "Get Date/Time String" VIs.
- *Get ROI from Image Display*[108] – Get the ROI (region of interest) from the main image display with a "Property Node."
- *Get ROI from Image Display or Default*[109] – Get the ROI (region of interest) from the main image display with a "Property Node." Use a default ROI defined by the Vision Assistant script when no ROI is defined on the image display.

[101] `http://youtu.be/rsI7fco5n_g` (2:47)
[102] `http://youtu.be/lEKJcR553zA` (2:48)
[103] `http://youtu.be/amunze4umlg` (2:30)
[104] `http://youtu.be/b5YCZdhHMGM` (7:45)
[105] `http://youtu.be/l6HWwnL_9zI` (1:28)
[106] `http://youtu.be/15V-gJwRI2Y` (3:26)
[107] `http://youtu.be/U4aNrolgDMA` (1:02)
[108] `http://youtu.be/sKeO2iM8UWE` (2:29)
[109] `http://youtu.be/p1sm8iVpVfo` (2:59)

- *Get Compound ROI from Image Display*[110] – Define a compound ROI (region of interest) on the main image display, and then retrieve the ROI with a "Property Node."
- *Select Image Palette*[111] – Use a Property Node to set the image display palette to either "Binary" or "Grayscale."
- *myRIO Analog Output Express VI*[112] – Place and configure the NI myRIO Analog Output Express VI.
- *myRIO Onboard Button*[113] – Place the NI myRIO onboard button Express VI.
- *Create Formatted Text String*[114] – Create a formatted text string with the "Format Into String" VI. The syntax is similar to the "printf" function in C.
- *Create "Flag Change" SubVI*[115] – Create the "Flag Change" subVI to detect when a Boolean flag changes state. This tutorial shows the complete process to create a subVI: define the front-panel controls and indicators, wire the block diagram, draw the icon, connect the terminals, save the subVI to the project folder, and place the subVI into the caller VI. Because this subVI contains a feedback node (state memory), select the "Preallocate clone reentrant execution" mode to allow multiple instances of the subVI to work properly.
- *Create an X-Y Plot*[116] – Create an X-Y plot from two arrays.
- *Create "Coin Info" Cluster Array*[117] – Create a "Coin Info" cluster array type definition to store each coin's value and name as well as the text label foreground and background colors for the nondestructive overlay.
- *Create "Inventory" Cluster of Arrays*[118] – Create a cluster of arrays for the Point-Of-Sale (POS) Terminal "inventory" control.

[110] `http://youtu.be/a8UVbpmXxjM` (2:37)
[111] `http://youtu.be/M53-QmPsSq4` (2:09)
[112] `http://youtu.be/gaDlExM4WbU` (2:06)
[113] `http://youtu.be/Sa4SsFWgVew` (0:57)
[114] `http://youtu.be/VqH0SSK1fHY` (1:44)
[115] `http://youtu.be/N1hUkWeZ8x0` (6:58)
[116] `http://youtu.be/TnKXqOsjyF0` (1:56)
[117] `http://youtu.be/Yqv4u5FE8Vk` (5:12)
[118] `http://youtu.be/DJB1otFHnbA` (5:03)