VIDEO SURVEILLANCE*

Jacob Fainguelernt

This work is produced by OpenStax-CNX and licensed under the Creative Commons Attribution License 2.0†

Abstract

This module describes the design flow of a motion detection system based video surveillance system based on the DM6437 Digital Video Development Platform (DVDP). The system will be able to identify and record only “interesting” video frames containing motion.

1 Introduction

Continuous-scene monitoring applications, such as ATM booths, parking lots or traffic monitoring systems, generate large volumes of data. Recording and archiving such volumes of data is a real problem, and one way to solve this is to reduce the size of the data stream right at the source.

In addition to traditional methods for compressing individual video images, we could identify and record only “interesting” video images, such as those images with significant amounts of motion in the field of view. That could significantly help reduce the data rates for surveillance-specific applications.

In this section, we’ll illustrate the design flow of a motion detection system based video surveillance system. The system will be able to identify and record only “interesting” video frames containing motion.

The development process will follow the following three steps:

1. Functional Requirements
2. System modeling and simulation
3. Code generation and implementation

1.1 Hardware and Software Requirements

This laboratory was originally developed using the following hardware and software:

- MATLAB® R2008a
- Code Composer Studio (CCS) v3.3
- Texas Instruments DM6437 hardware.

*Version 1.1: Jun 4, 2009 6:10 am -0500
†http://creativecommons.org/licenses/by/2.0/
1.2 Related Files

- Powerpoint Presentation - Surveillance.ppt
- Simulink Model for Simulation - surveillance.mdl
- Video-clip viplane.avi
- MATLAB script my_tcpip_surveil_script.m
- Simulink Model for Real-Time (DM6437) - my_dm6437evmsurveillance_tcpip.mdl
- Simulink Model for Real-Time Lane Detection (PC) - my_pc_surveillance_tcpip.mdl

2 Video Surveillance

In this session we will show how to create the line detection model, and how it can be integrated in Simulation and Real-Time Implementations.

![Figure 1: The Motion Detection Process](http://cnx.org/content/m24470/latest/)

2.1 The Motion Energy Estimation

The Motion Energy estimation is based on the calculation of the Sum of Absolute Differences (SAD) according to the following equation:

\[ E = \sum_{i,j} |I(x,y) - I(x,y + 1)| \]

See the file at <http://cnx.org/content/m24470/latest/Surveillance.ppt>
See the file at <http://cnx.org/content/m24470/latest/surveillance.mdl>
See the file at <http://cnx.org/content/m24470/latest/viplane.avi>
See the file at <http://cnx.org/content/m24470/latest/my_tcpip_surveillance_script.m>
See the file at <http://cnx.org/content/m24470/latest/my_dm6437evmsurveillance_tcpip.mdl>
See the file at <http://cnx.org/content/m24470/latest/my_pc_surveillance_tcpip.mdl>
\[
\text{SAD} = \sum_i \sum_j | I_k(i,j) - I_{k-1}(i,j) | \\
\text{where:}
\]

\begin{tabular}{|c|c|}
\hline
$I_k(i,j)$ & - Current Frame \\
$I_{k-1}(i,j)$ & - Previous Frame \\
\hline
\end{tabular}

Table 1

2.2 Functional Description

The Video Surveillance system block diagram is shown in Figure 2. The source video is displayed and played back through the system. This incoming video stream is processed by a subsystem that estimates the motion within the scene and captures the interesting video frames. The system can display either the recorded video frame or the Absolute Differences (AD) image.

The user can configure the motion energy threshold value and select which image will be displayed (Display Control). The system outputs include, besides the displayed image, the following features:

- A graph of the motion energy as a function of time. This graph displays the threshold value as well.
- The Frame count of the recorded image
- An output signal triggered by motion detection (Trigger)

While the generated code is running on the target, a host side Simulink® model simultaneously sends video frames to the target via TCP/IP protocol using TCP/IP Blocks from Instrument Control Toolbox™. The target receives video frames sent by the host side Simulink model, computes the sum of the absolute value of differences (SAD) between successive video frames, and returns an estimate of motion. When the motion estimate value exceeds a threshold, the target increments a counter and sends the corresponding frame back to the host.
3 Simulation

1. Open a new Simulink model
2. Add the blocks shown in the following table (Please refer also to Figure 3):

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Block</th>
<th>Blockset</th>
<th>Quant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAD</td>
<td>Subsystem</td>
<td>Simulink / Ports &amp; Subsystems</td>
<td>1</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Action</th>
<th>Block</th>
<th>Library</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay</td>
<td></td>
<td>Signal Processing Blockset / Signal Operations</td>
<td>1</td>
</tr>
<tr>
<td>Abs</td>
<td></td>
<td>Simulink / Math Operations</td>
<td>1</td>
</tr>
<tr>
<td>Matrix Sum</td>
<td></td>
<td>Signal Processing Blockset / Math Functions / Matrices and Linear Algebra / Matrix Operations</td>
<td>2</td>
</tr>
<tr>
<td>Data Type Conversion</td>
<td></td>
<td>Simulink / Signal Attributes</td>
<td>1</td>
</tr>
<tr>
<td>Add Sum,</td>
<td></td>
<td>Simulink / Math Operations</td>
<td>1</td>
</tr>
<tr>
<td>Recording</td>
<td>Enabled Subsystem</td>
<td>Simulink / Ports &amp; Subsystems</td>
<td>1</td>
</tr>
<tr>
<td>Add Sum,</td>
<td></td>
<td>Simulink / Math Operations</td>
<td>1</td>
</tr>
<tr>
<td>Delay</td>
<td></td>
<td>Signal Processing Blockset / Signal Operations</td>
<td>1</td>
</tr>
<tr>
<td>counter</td>
<td>Subsystem</td>
<td>Simulink / Ports &amp; Subsystems</td>
<td>1</td>
</tr>
<tr>
<td>Add Sum,</td>
<td></td>
<td>Simulink / Math Operations</td>
<td>1</td>
</tr>
<tr>
<td>Delay</td>
<td></td>
<td>Signal Processing Blockset / Signal Operations</td>
<td>1</td>
</tr>
<tr>
<td>Generate data for motion energy and threshold display</td>
<td>Matrix Concatenate</td>
<td>Simulink / Math Operations</td>
<td>2</td>
</tr>
<tr>
<td>Compare Motion Energy against Threshold</td>
<td>Relational Operator</td>
<td>Simulink / Logic and Bit Operations</td>
<td>1</td>
</tr>
<tr>
<td>Input Video</td>
<td>From Multimedia File</td>
<td>Signal Processing Blockset / Signal Processing Sources</td>
<td>1</td>
</tr>
<tr>
<td>Display Control</td>
<td>Constant</td>
<td>Signal Processing Blockset / Signal Processing Sources</td>
<td>1</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Display Motion Energy and Threshold</th>
<th>Floating Scope</th>
<th>Simulink/Sinks</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draws formatted text on an image or video stream</td>
<td>Insert Text</td>
<td>Video and Image Processing Blockset / Text &amp; figure</td>
<td>1</td>
</tr>
<tr>
<td>Connected to the trigger (not used for simulation)</td>
<td>Terminator</td>
<td>Simulink/Sinks</td>
<td>1</td>
</tr>
<tr>
<td>Images’ Display</td>
<td>To Video Display</td>
<td>Video and Image Processing Blockset / Sinks</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2

Figure 3: Simulink Blocks for Video Surveillance

3.1 Configuring the SAD blocks

1. Configure one of the “Add” block for subtraction, you may also change its icon shape to a circular form as follows:\footnote{The symbol \texttt{\(\mid\)} is used to place the \texttt{\(\pm\)} and \texttt{\(\pm\)} perpendicular to each other}:

http://cnx.org/content/m24470/1.1/
2. Configure its fixed point mode of operation Configure one of the “Matrix Sum” blocks for adding rows:
3. Configure its fixed point mode of operation
4. Configure the fixed point mode of operation for the second Matrix “Column Add” block:
3.2 Configuring the Input blocks

1. Configure one of the “Video Source” "From Multimedia File" block for reading frames from the "vimp-men.avi" file.
2. Set the initial value of the "Constant" to 1e5.
3. Configure "$Data Type Conversion" block:

Figure 9

Figure 10
4. Configure "Relational Operator" block:

![Relational Operator Block Diagram]

Figure 11

5. Configure one of the "Matrix Concatenate" block:

![Matrix Concatenate Block Diagram]

Figure 12
6. The “Floating Scope” will be used to display the motion energy and the threshold\(^8\) as a function of time. Double-Click the block. A cope screen will appear, you should then click the icon as shown in the following picture:

![Image of floating scope configuration](http://cnx.org/content/m24470/1.1/)

**Figure 13**

You may keep the default configuration for "Data history".

7. Configure the "Insert Text" block:

---

\(^8\)The threshold is changed upon a user command. The simultaneous display of the threshold and motion energy will enable the user to follow-up the process of recording frames.
8. Double-click on the "Enabled Subsystem" block, and build it as shown below:
9. Double-click on one of the "Subsystem" block and build it as shown below:

Figure 16

Save the subsystem as: "Motion Energy"

10. Double-click on one the remaining "Subsystem" block, build it as shown below and name it "Frame Counter":

Figure 16
3.3 Connecting the blocks:

1. Connect the blocks as shown in the following figure:

![Figure 18: Video Surveillance model](http://cnx.org/content/m24470/1.1/)
3.4 Running the model

1. Run the model. You should observe the following images:

<table>
<thead>
<tr>
<th>Current Frame</th>
<th>Recorded Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Current Frame Image" /></td>
<td><img src="image2" alt="Recorded Frame Image" /></td>
</tr>
</tbody>
</table>

Table 4

2. Double click the scope and you will see:
Figure 19

In this scope display, yellow represents the motion estimate from each video frame, and magenta is the threshold level. In the Motion Threshold scope here, we see that we identified five frames with motion above the threshold that were captured and recorded.

4 Real Time Implementation

4.1 Video Surveillance using TCP/IP

1. Open the “video_sim.mdl” Simulink model (generated in the "A Framework for Video Processing with the DM6437 DVDP” module).
2. Rename the model “my_dm6437evmsurveil_tcpip.mdl”.
3. Rename the “R_W Algorithm” subsystem to “Surveillance Algorithm”.
4. Double-click the “Surveillance Algorithm” block.

http://cnx.org/content/m24470/1.1/
5. In this block, add a Function-Call Subsystem from the "Ports & Subsystems" group of the "Simulink" Blockset.
6. Rename “Function-Call Subsystem” to “Video Frame Capture Motion Threshold” and Double-click it.
7. Open the model "surveillance_sim.mdl", created in section and copy it into “Video Frame Capture Motion Threshold”.
8. Add three input ports "In1" from the "Sources" group of the "Simulink" blockset and three output ports "Out1" from the "Sinks" group of the "Simulink" blockset.
9. Add a "Switch" from the "Signal Routing" group of the "Simulink" blockset and a "Gain" from the "Math Operation" group of the "Simulink" blockset.
10. Double-click the "Motion Energy" block and add him another output port as shown:

Figure 20

11. Connect the blocks as shown:
12. Configure the "Gain" block:

Figure 21

Figure 22
13. Configure the "Switch" block:

![Switch block configuration](http://cnx.org/content/m24470/1.1/)

**Figure 23**

14. Double-click the “Surveillance Algorithm” subsystem.

15. Configure the "Byte Unpack" block:

![Byte Unpack block configuration](http://cnx.org/content/m24470/1.1/)

**Figure 24**

16. Configure the "Byte Pack" block:
17. Configure the "TCP/IP Receive" block:

**Figure 25**

**Figure 26**
18. Configure the "TCP/IP Send" block:

![TCP/IP Send Block Configuration](http://cnx.org/content/m24470/1.1/)

**Figure 28**

19. Connect the "Video Frame Capture Motion Threshold" block as shown:

![Video Frame Capture Motion Threshold Block](http://cnx.org/content/m24470/1.1/)
20. Open the "RW_pcl_tcpip.mdl" model, created in the "Video R_W" file
21. Rename the model "my_pc_surveil_tcpip.mdl".
22. Add the "Manual Switch" from the "Signal Routing" group of the "Simulink" blockset and three "Constant" blocks from the "Signal Processing" group of the "Signal Processing" blockset.
23. Configure the "Constant" blocks:
24. Configure the "Byte Unpack" block:

![Function Block Parameters: Byte Pack](image)

**Figure 30**

25. Connect the block as shown:

http://cnx.org/content/m24470/1.1/
26. Configure the "TCP/IP Receive" block:
27. Double-click the “Display” subsystem.
29. Configure the "Byte Unpack" block:

![Function Block Parameters: Byte Unpack](http://cnx.org/content/m24470/1.1/)

30. Connect the block as shown:
31. Add to the "Display" subsystem the "Insert Text" from the "Text & figure__" group of the "Video and Image Processing" blockset, "Floating Scope" from the "Sinks" group of the "Simulink" blockset.
32. Configure the "Insert Text" block:
33. Add the “Subsystem” from the “Ports & Subsystems” group of the "Simulink" Blockset
34. Rename it "Combine Count"
35. Add a "Frame Counter" subsystem as we did in paragraph 14 at the first section.
36. Add a "Matrix Concatenate" from the "Math Operations" group of the "Simulink" blockset.
37. Connect the block as shown:

38. Configure the "Matrix Concatenate" block:
39. Connect the "Display" subsystem as shown:
40. Open a new m-file, copy the following code and save it as "my_tcpip_surveil_script.m" at the same directory as the model.

```matlab
function my_tcpip_surveil_script(hostModelName)
%my_tcpip_surveil_script controls host-side TCP/IP processing for
% Video Surveillance demo:
% 1) Builds and runs the target application
%********** Initialization ****************
% Get model name
modelName = gcs;

% Connect to CCS
CCS_Obj = connectToCCS(modelName);
saved_visibility = CCS_Obj.isvisible;
CCS_Obj.visible(1);

% Load application
loadApp(modelName, CCS_Obj);

% Run application
fprintf('Running application: %s
', modelName);
CCS_Obj.run;

% Allow some time for DHCP address acquisition
pause(3);

% Retrieve target's host name
```
boardType = get_param([modelName '/IP Config'], 'boardType');
userPrompt = sprintf('Enter the IP address or the host name of the %s board: ', boardType);
hostName = inputdlg(userPrompt, 'Target IP address');
if isempty(hostName)
    errordlg('You have to provide a valid IP address or host name to run the demo.', ...
            'TCP/IP Surveillance Recording', 'modal');
    return;
end
hostName = strtrim(hostName{1});

% Launch host side UDP receive / Video display model
fprintf('Launching host side application: %s %n', hostModelName);
open_system(hostModelName);

% Update host side TCP/IP blocks with target's IP address
set_param([hostModelName '/TCP/IP Send'], 'Host', hostName);
set_param([hostModelName '/TCP/IP Receive'], 'Host', hostName);
fprintf('Click on play button to start running the host side application!%n\n\n')

%EOP tcpip_surveil_script.m

41. Open a new m-file, copy the following code and save it as "my_tcpip_surveil_script.m" at the same
directory as the model.
42. now we have two models ("my_pc_surveil_tcpip.mdl" and "my_dm6437evmsurveil_tcpip.mdl") and
one MATLAB file ("my_tcpip_surveil_script.m")

4.2 Running the Real-time Model

1. Build load and run the project from the "my_dm6437evmsurveil_tcpip.mdl" model. Double-click the
" Build / Reload & Run" block.
2. Running the script, you will see:

![Image of original video and motion frames]

Figure 40

http://cnx.org/content/m24470/1.1/
If you want to display the Absolute Differences Image, change the state of the switch from 0 to 1. You will then see:

![Figure 41](http://cnx.org/content/m24470/1.1/)

![Figure 42](http://cnx.org/content/m24470/1.1/)