Decision-Tree Based Pixel Classification for Real-time Citrus Segmentation on FPGA

Reconfig 2019

Ismael Antonio Dávila Rodríguez, Marco Aurelio Nuño Maganda, Yahir Hernandez Mier, Said Polanco Martagón

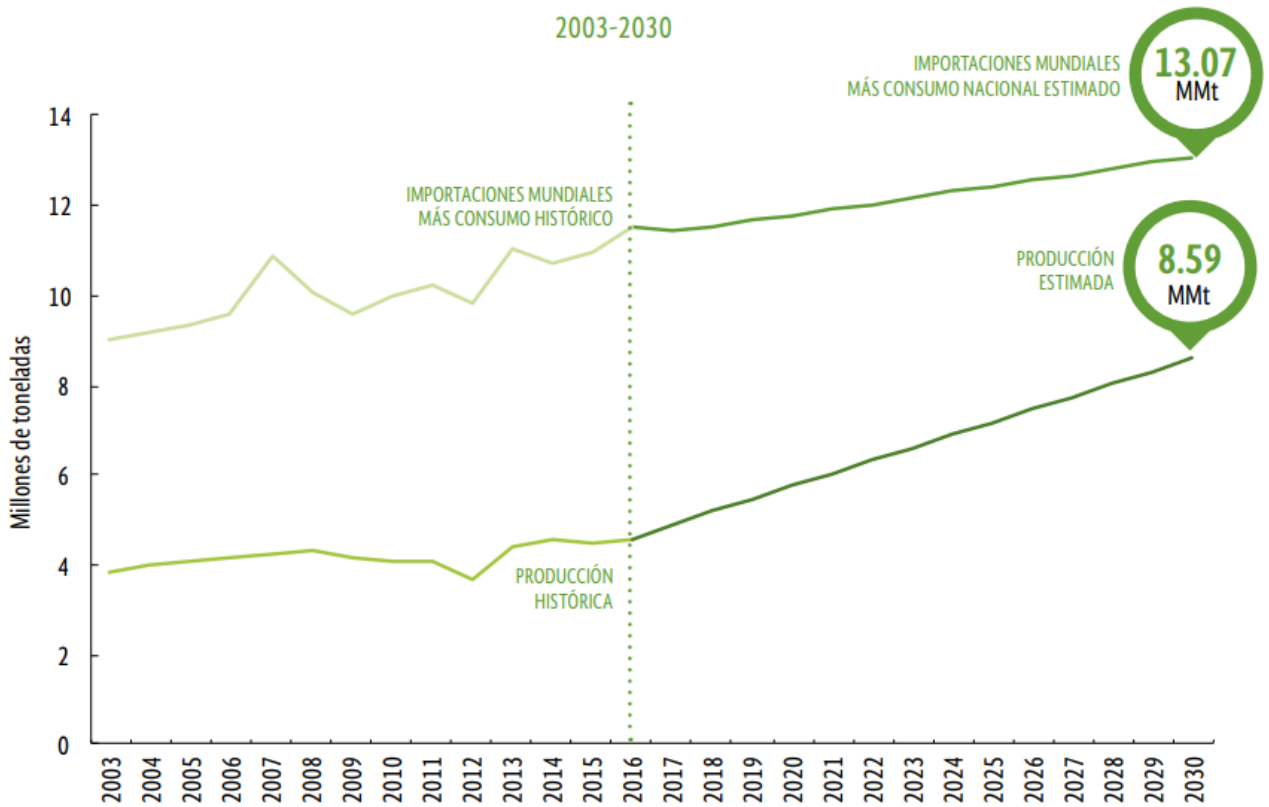
Universidad Politécnica de Victoria

December 10th of 2019

Abstract

According to Food and Agriculture Organization, Mexico is one of the top 5 citrus producers in the world. In order to accomplish the required quality control to export their products, citrus producers require sorting machines able to classify millions of fruits according to their size, color and defects. Computer vision provides image processing tools, as image segmentation, that could be used as first stage in a classification process. Fruit classification must be a fast process in order to be able to process as much fruits per second as possible. In this paper, an FPGA architecture for image segmentation of orange images using decision-tree models is proposed. A decision-tree model is proposed because global thresholding fails due to the noise produced by moving fruits in a classification line, and adaptive thresholding algorithms are not suitable for real-time applications, because of high computing power and memory requirements. A decision-tree model requires less hardware compared to mentioned traditional algorithms, and they can perform real-time segmentation because they are based on pixel serialization, and not on processing pixel neighborhoods. The proposed architecture was implemented in a Spartan-6 FPGA and an accuracy of 97.1% of correct segmented pixels, compared to an offline purposed segmentation, was achieved, running at 60 fps.

Introduction



Fuente: Elaboración propia con datos del SIAP y el SIAVI, 2017.

Figure: International Orange consumption / Mexico Orange Production expectation.

According to Mexico's Ministry of Agriculture and Rural Development (SAGARPA).

Color conversion

Usually in all digital systems the color space used for image representation is RGB space.

RGB to Grayscale:

$$Y'_{709} = 0.2126R' + 0.7152G' + 0722B' \tag{1}$$

RGB to HSV:

$$rng = V - min(R, G, B) \tag{2}$$

$$H = \begin{cases} \frac{60(G-B)}{2(rng)} & \text{if } V = R \\ \frac{120+60(B-R)}{2(rng)} & \text{if } V = G \\ \frac{240+60(R-G)}{2(rng)} & \text{if } V = B \end{cases} \tag{3}$$

$$S = \begin{cases} \frac{255(rng)}{V} & V \neq 0 \\ 0 & otherwise \end{cases} \tag{4}$$

$$V = max(R, G, B) \tag{5}$$

Morphological operations

(a) Image segmentation input.

(b) Erosion.

(c) Dilation.

Figure: Morphological operators example.

These operators are useful in image segmentation porosity.

Morphological operations

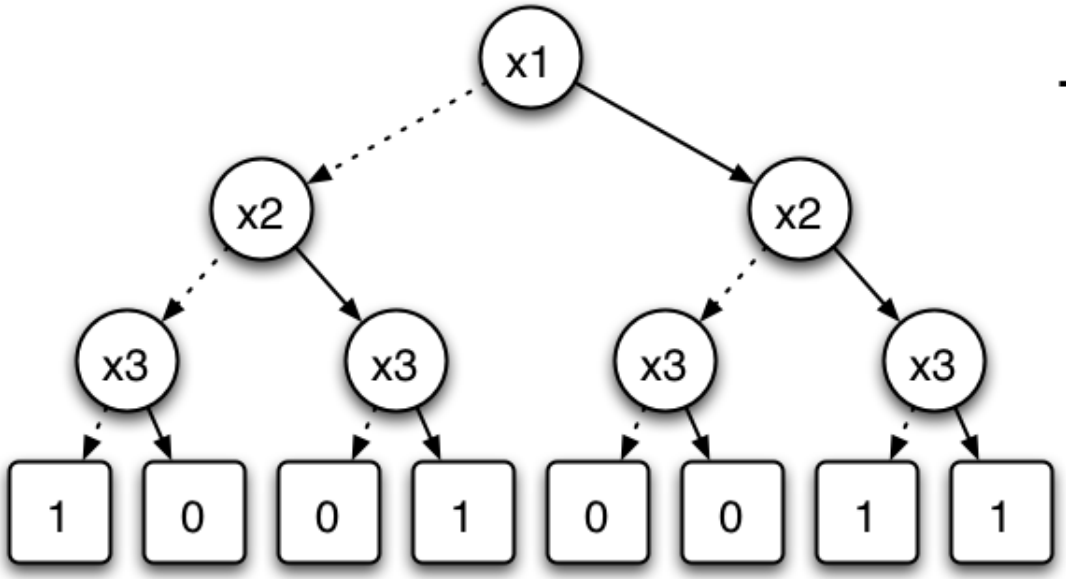
Figure: Closing operation.

This operation is equal to perform a Dilation followed by a Erosion.

Decision Tree Model

Decision trees (DTs) is a powerful and popular tool for classification and prediction . DTs are rooted tree structures, with leaves representing classifications and nodes representing tests of features that lead to those classifications. A tree can be “learned” by splitting the source set into subsets based on an attribute value test. Some popular algorithms for DT training are: CART, ID3, C4.5 and Random Forest.

Decision Tree (DT)



x1	x2	x3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Figure: Binary Decision Tree Example.

- Easy hardware implementation (only uses comparators).
- No arithmetic operations.
- No resource expensive.
- Can be solved as a parallel circuit.

Proposed Architecture

The main components of the proposed system are shown. An industrial camera takes input video and transmits to FPGA using a HDMI cable. The camera is located inside a dark cabin with a rail, which provides a controlled environment (constant illumination and black background). On the rail, oranges are transported and passed in front of the camera at a speed of 4 **oranges per second** (ops).

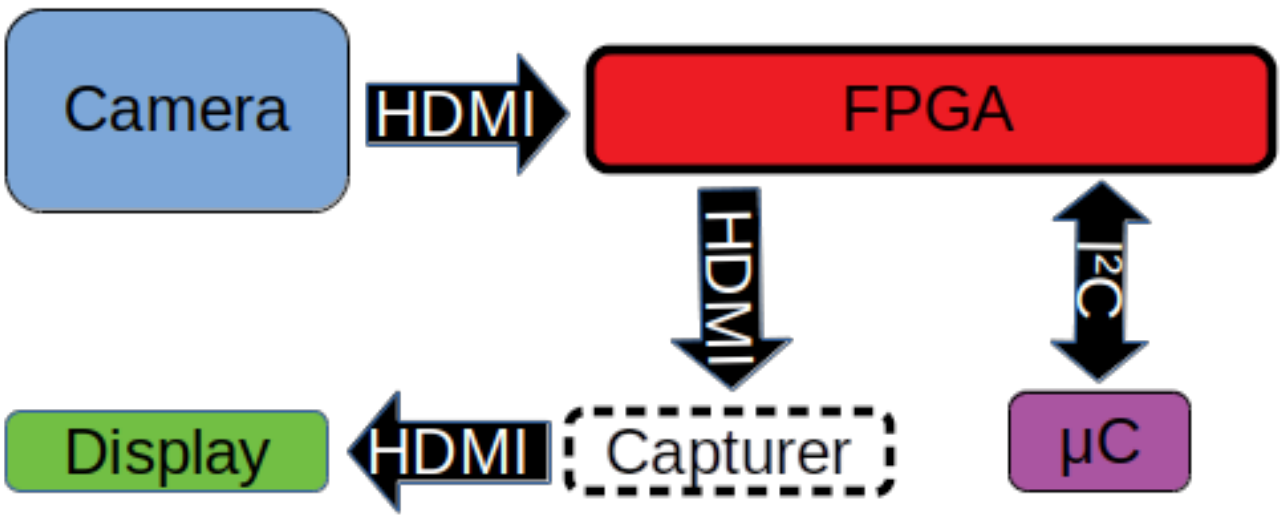


Figure: System elements overview.

Project platform

Figure: Orange transportation system.

Dark cabin

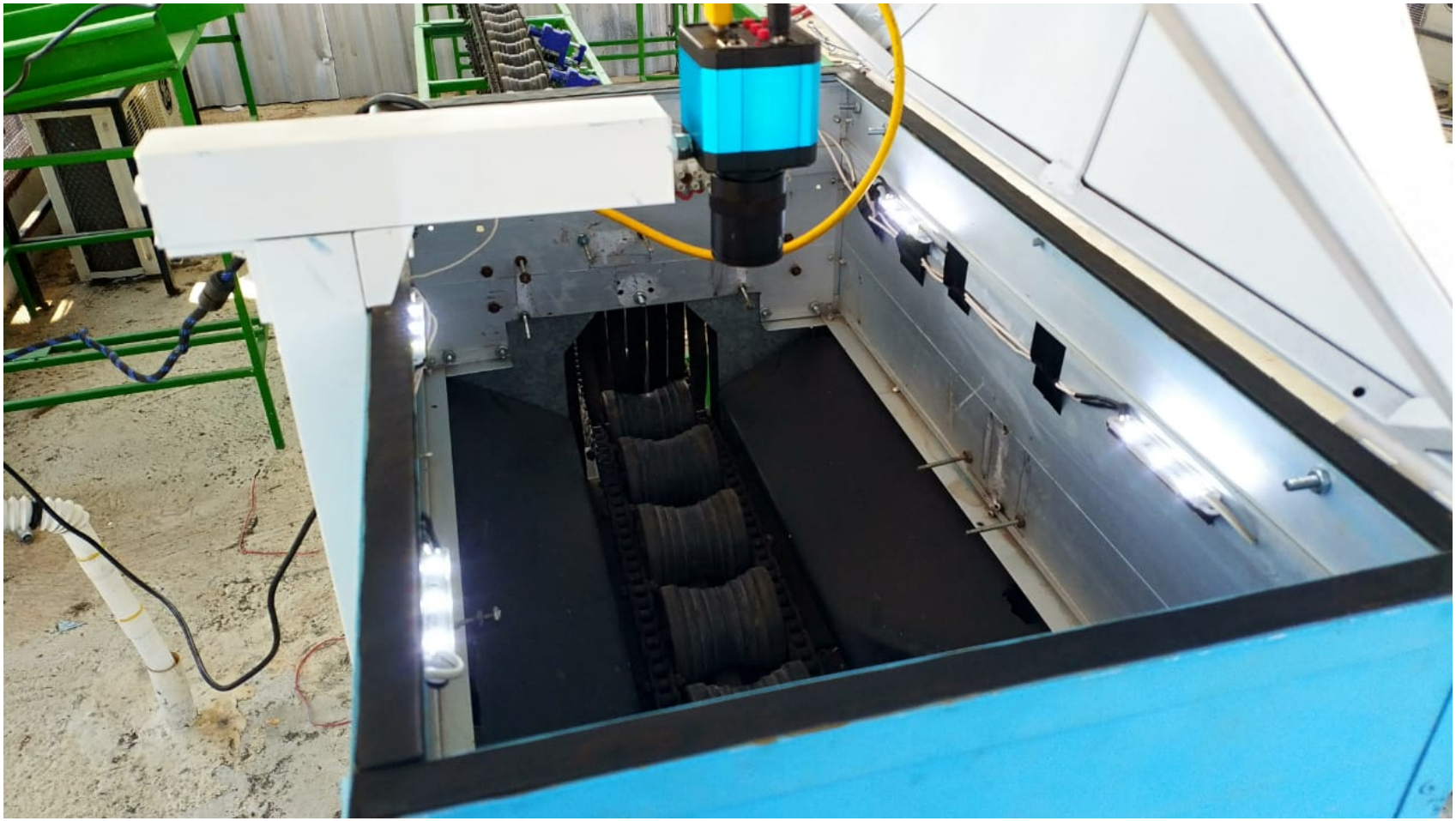


Figure: Dark cabin setup.

Project methodology

To preform this project the following methodology was implemented:

a) Decision Tree model Build
   1. Data set acquisition (Orange images)
   2. Feature extraction (R, G, B, H, S, V, Gray)
   3. Feature relevance
   4. Model generation (Training and Testing)

b) Hardware modules
   1. Pixel buffer
   2. HSV conversion
   3. Grayscale conversion
   4. Segmentation
   5. Morphological operations
   6. Mask drawer

c) Integration
   1. FPGA segmentation recording
   2. Segmentation comparison and measurements

Data set acquisition procedure

1. Record video
2. Region of interest (ROI) cropping
3. OpenCV segmentation
4. Center detection
5. Automated orange image acquisition
6. Manual image classification
7. Feature extraction (HSV and Grayscale conversion)
8. Pixel data set acquisition (Segmentation vs Features)

Data set acquisition

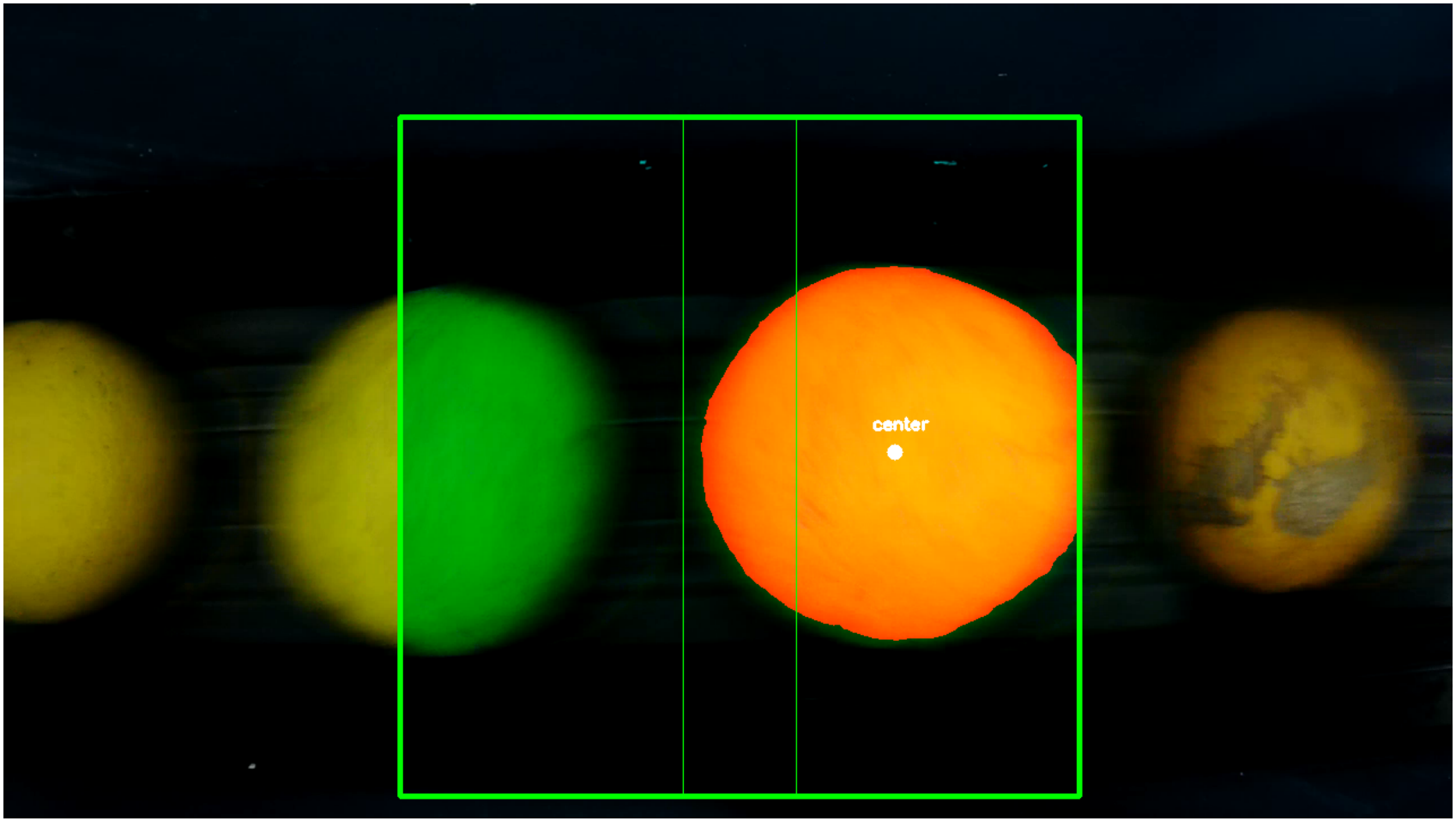


Figure: Video frame processing.

Feature extraction

Once the images of oranges were properly classified by size and color each image is converted to HSV and Grayscale color spaces for taking the 7 features and build the DT model.
The data set is compiled with the 7 feature and the pixel in segmentation as class (segmented pixel / non-segmented pixel).

Feature relevance

A feature analysis to obtain the best features for building the model was performed. The following analysis were used:

1. Relative Absolute Error (RCA).
2. Correlation Attribute analysis (CAA).
3. Info Gain Attribute Evaluation (IGAE).
4. Gini impurity (GI)
5. Entropy (E)

Model generation

The decision tree HDL module was generated using the Decision Tree HDL Builder script (DTHB) that converts the decision tree model behaviour to a synthesizable HDL module. This was done by a text parser script written in python.

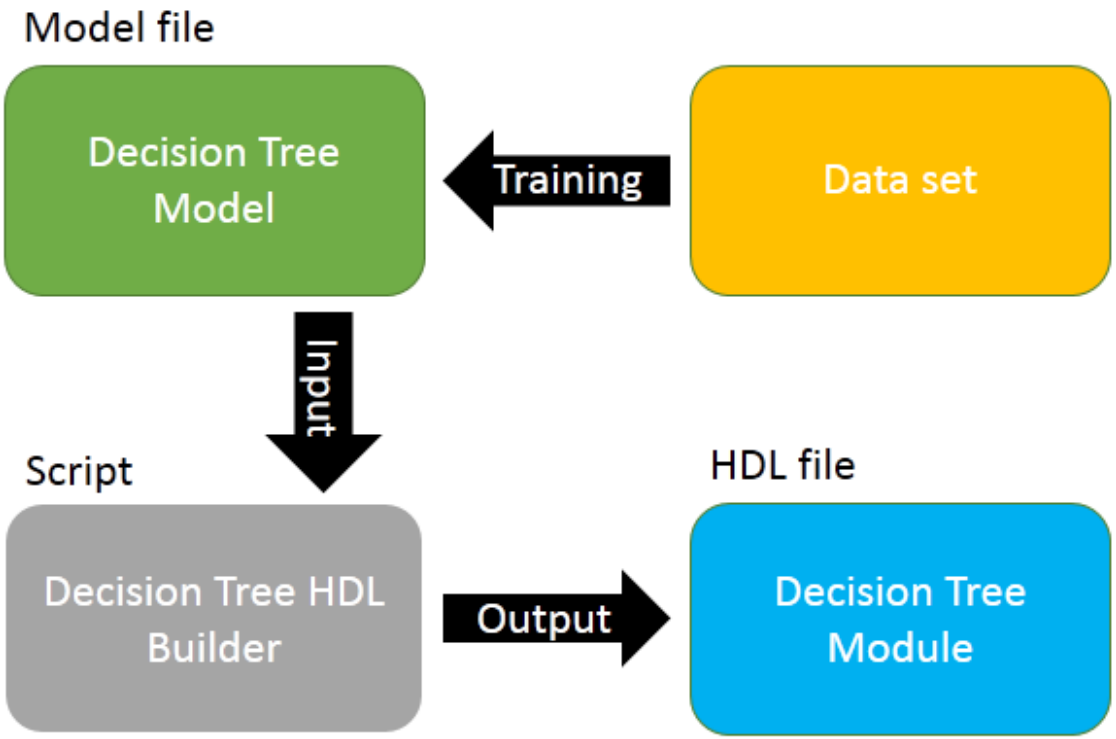


Figure: Decision Tree module generator.

Hardware overview

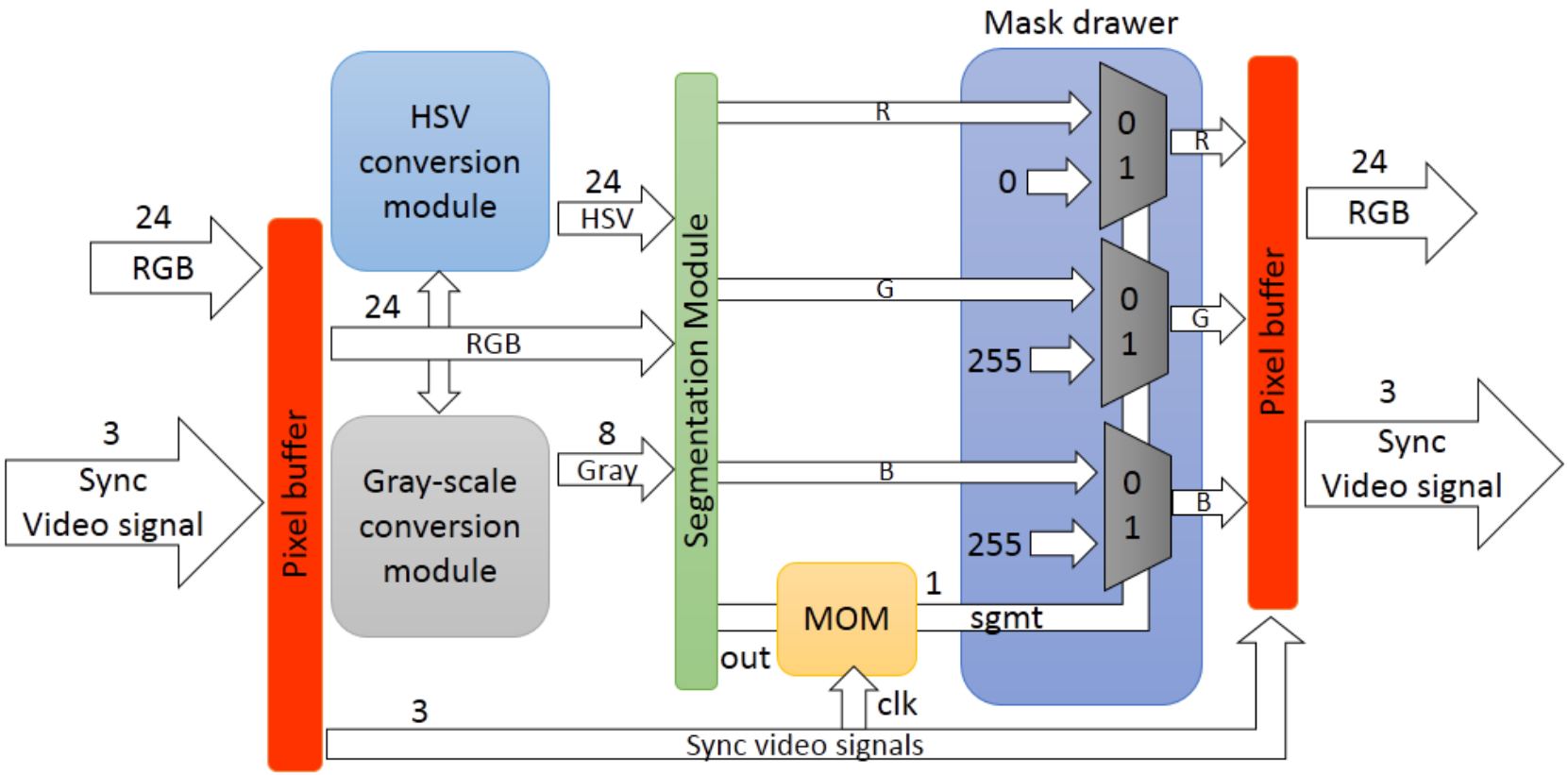


Figure: Data flow architecture.

Module description

- **Pixel buffer module**
  This module handles *RGB* bus and synchronization signals. All synchronization signals are buffered in this stage to prevent signal deformation. *RGB* bus is stored in a register to be able to read by any other modules.
- **HSV conversion module**
  This module makes a conversion from *RGB* to HSV color space and is based on the OpenCV documentation. It is a combinational circuit that calculates each components.
- **Gray scale conversion module**
  As mentioned in a previous section, to calculate gray-scale pixel was used equation 1. Fixed point arithmetic was used to calculate the output.

Module description

- **Morphological operations module (MOM)**
  This module receives the segmented frame as input. Its main purpose is to eliminate porosity and noise in the segmented object. It consist in a Closing + Dilation morphological operations applied in series. Closing operation was performed using a 7×7 disk structuring element. Dilation was applied using a 3×3 disk structuring element. This module shifts segmentation 10 vertical and horizontal pixels (4 pixel for each 7×7 kernel and 2 more for 3×3 kernel). Clock video signal is needed to synchronize the register where the segmentation is stored.
- **Mask drawer module**
  Once the segmentation is done, the border of the segmented structure is drawn in cyan color over the input frame. This can be done by the multiplexers. When the module finds a segmented pixel, it prints a cyan pixel, otherwise prints the value of the input pixel. The purpose of marking these border pixels is to make easier for the FPGA to segment the structure and to synchronize videos to make an I/O video comparison.

Segmentation module

The module was generated with Decision Tree HDL Builder script (DTHB). The algorithm only can solve decision trees with binary nodes, also the output only can be binary (only can be classified 2 classes).
Implementing this aproach has the advantage of reducing and mantaining the combinational path to be the same in all of its parts. Also, this approach makes the tree depth to be less significative, since it only needs more comparators (that are less resourceless expensive), than adding comparators in series.

Segmentation module generation example

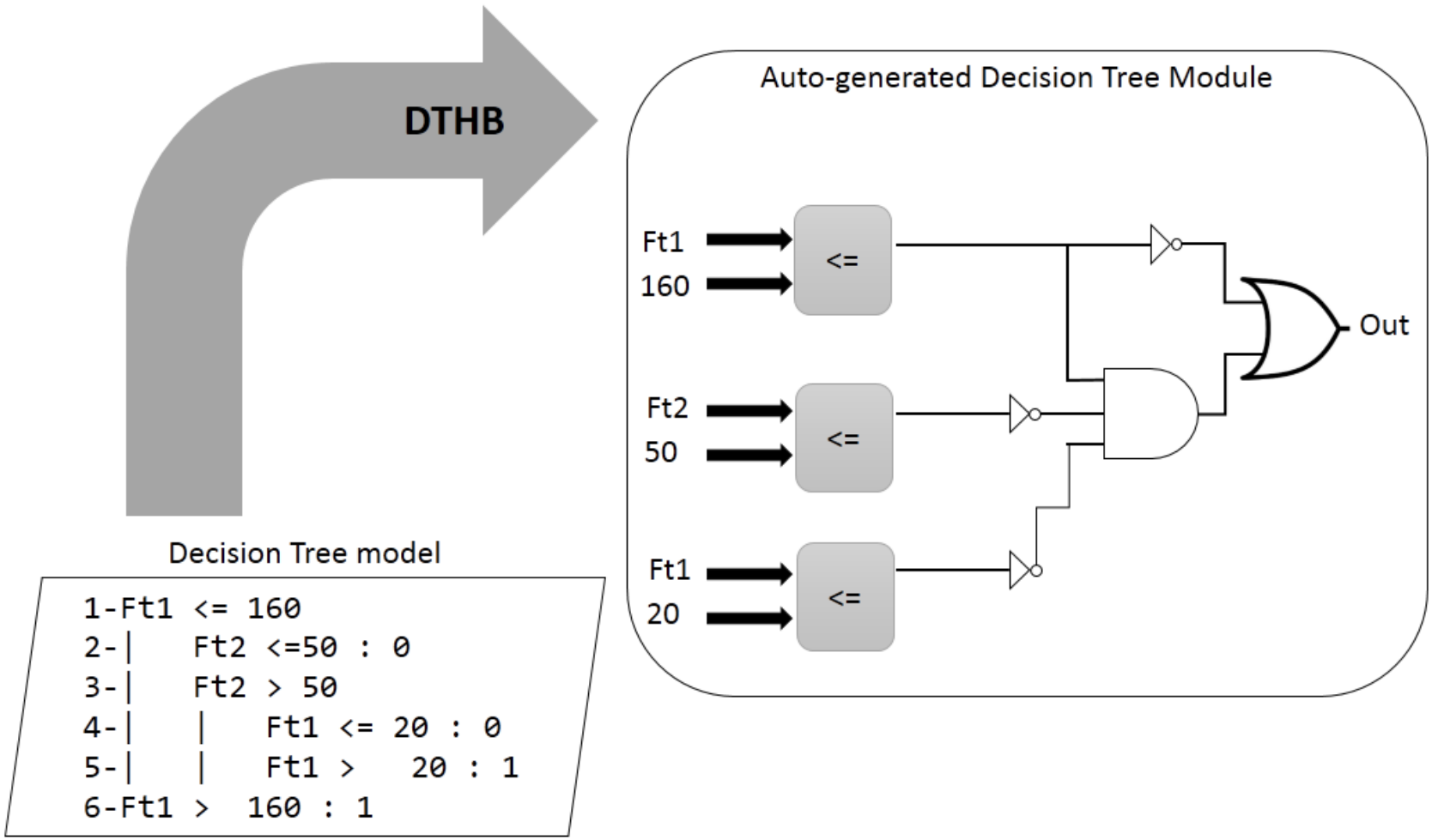


Figure: Example of decision tree hardware module.

Results and discussion

Resources.

The hardware elements used for the proposed system are:

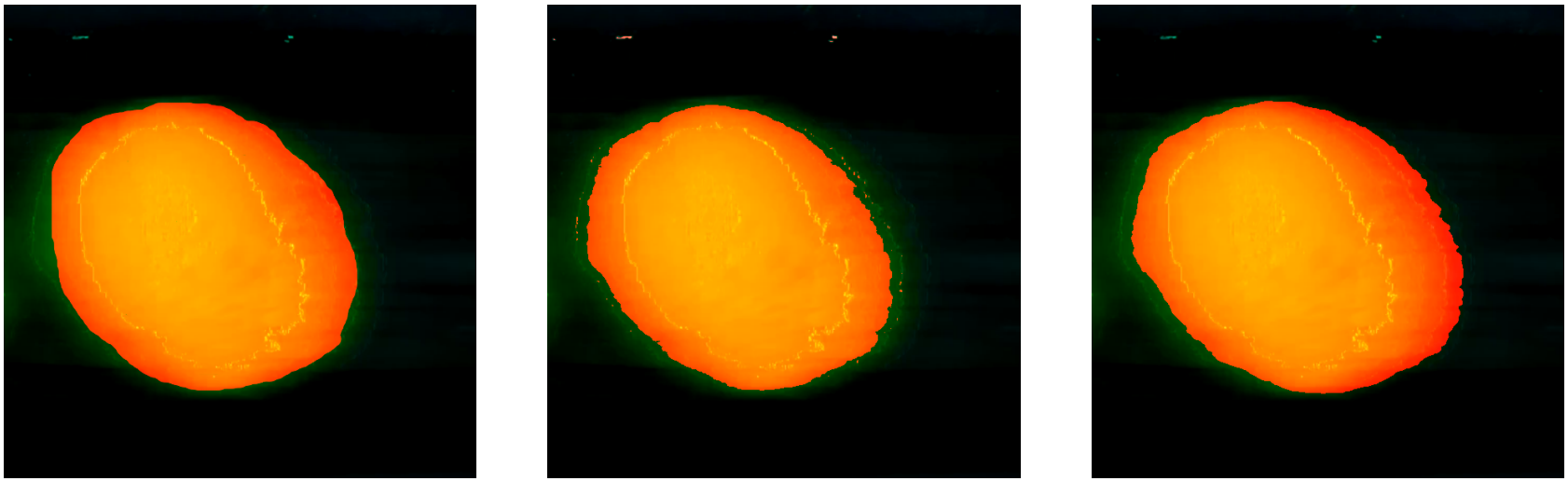
- Industrial Video Processing Kit (IVPK) Spartan6: is a FPGA development board with all hardware needed for image processing.
- Industrial camera YW2307: rough use camera for long active recording tasks with HDMI 1080p 60fps output.
- Atmega328: A microcontroller used for controlling all machine actuators.
- HDMI display: A regular TV with HDMI input video, only for debugging purposes.
- Avermedia-C875: A video card capturer for output video recording, used for result analysis.

Results and discussion

Resources.
The software used for the proposed system are:

- Weka(v3): used for feature analysis and model generator.
- Python(v3.6): used for text parsing, and feature analysis.
- OpenCV(v4): used for image processing and data acquisition.
- Xilinx ISE (v14.7): used for HDL synthesis and hardware debugging.

Thresholding algorithm comparison



(a) Manual image segmentation.

(b) Otsu adaptive thresholding result.

(c) Proposed segmentation.

Figure: Proposed segmentation algorithm.

The resulting accuracy was 96.6% for the Otsu algorithm and 97.3% for the proposed one. Proposed algorithm only takes the biggest segmented area.

Data set acquisition

Table: Generated data set details.

Resource	N. samples	Description
Video	3	Low speed (0.5 ops) (720p,60fps)
Video	3	Medium speed (2 ops) (720p,60fps)
Video	3	High speed (5 ops) (720p,60fps)
Images	1836	Oranges photos (600x600)
Images	1836	Segmented Oranges photos (600x600)
Pixels	136886	Pixel samples (7 features, 2 classes)

Feature importance analysis

Table: Feature relevance analysis.

Analysys/feature	R	G	B	H	S	V	Gray
RAE	0.0255	**0.0194**	0.0195	0.226	0.0262	0.0166	**0.0143**
CCA	0.3211	**0.5164**	0.113	0.2362	0.0489	**0.4733**	0.4652
IGAE	0.1363	**0.2596**	0.0696	0.2004	0.1210	0.2311	**0.2370**
GI	0.1211	**0.2031**	0.1024	0.1552	0.1111	0.1387	**0.1681**
E	0.1235	**0.1909**	0.1144	0.1598	0.1167	0.1332	**0.1611**

DT model training

Table: Tree models details.

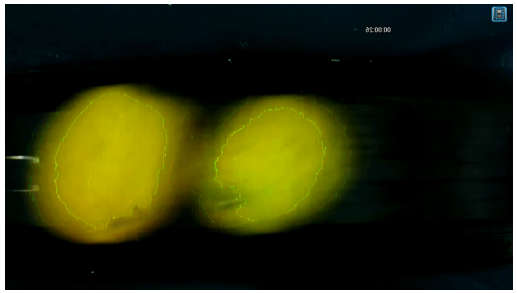
Model	Features	Validation type	Validation accuracy
M0	Gray & Hue	10-fold	83.24%
		70%-Test 30%-Train	83.17%
M2	Gray & Green	10-fold	81.67%
		70%-Test 30%-Train	81.50%
M4	All	10-fold	**83.52%**
		70%-Test 30%-Train	**83.41%**

Decision Tree models

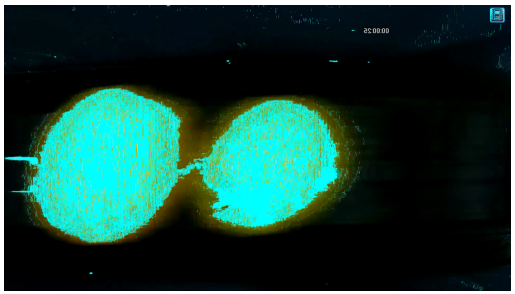
Table: Model segmentation descriptions.

Model	Composition	Kernel size
M0	None	None
M1	M0+Closing+Dilate	7×7
M2	None	None
M3	M2+Closing+Dilate	7×7
M4	None	None
M5	M4+Closing+Dilate	7×7

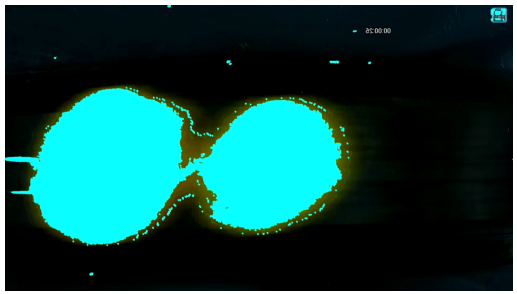
FPGA+DT model segmentation result



(a) Sample frame.



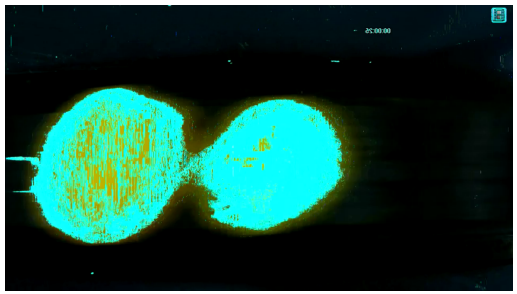
(b) $M0$ segmentation.



(c) $M1$ segmentation.

(d) $M2$ segmentation.

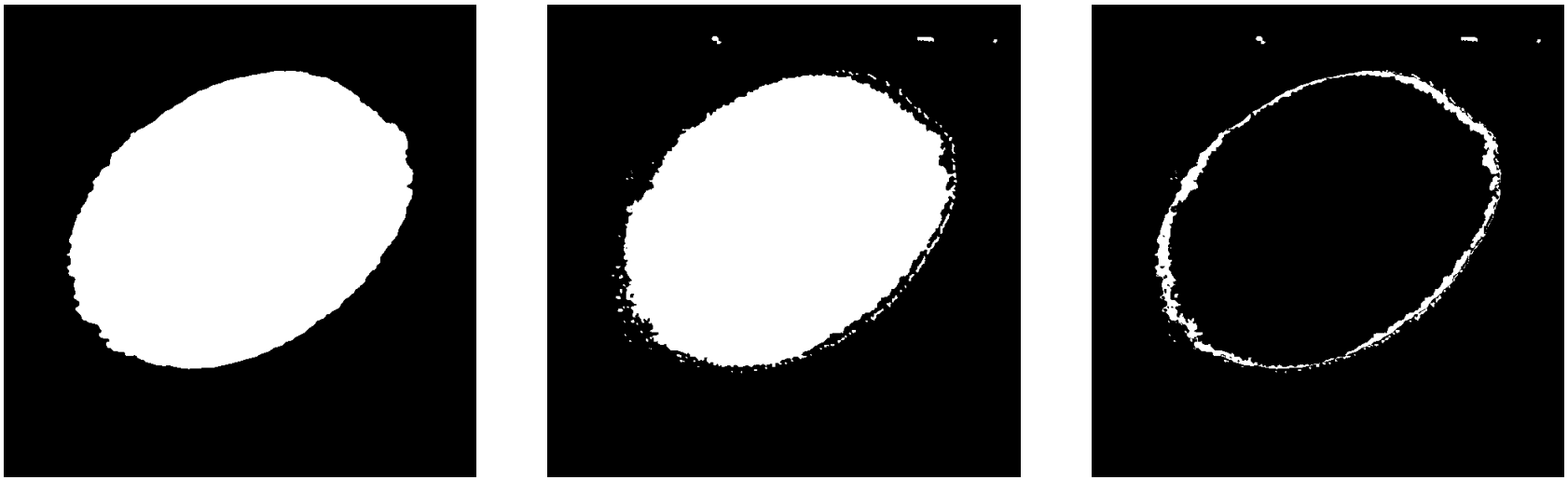
(e) $M3$ segmentation.



(f) $M4$ segmentation.

(g) $M5$ segmentation.

Thresholding algorithm comparison



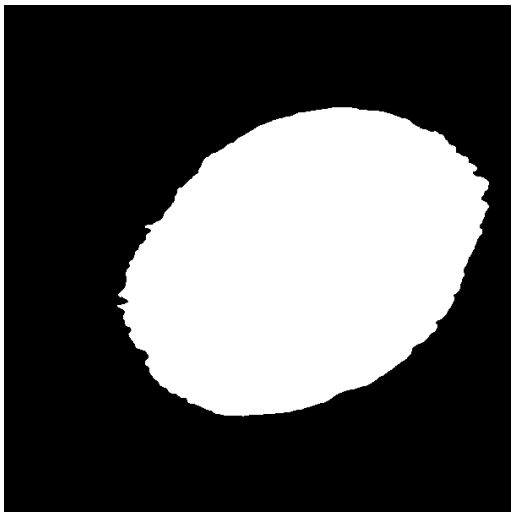
(a) Proposed segmentation.

(b) FPGA segmentation.

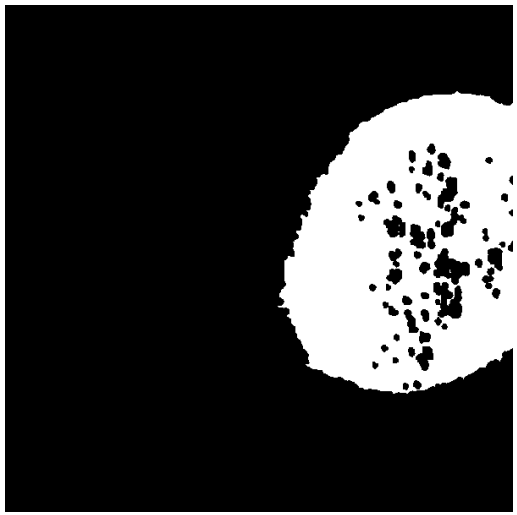
(c) Sample of differential segmentation.

Figure: Segmentation outputs.

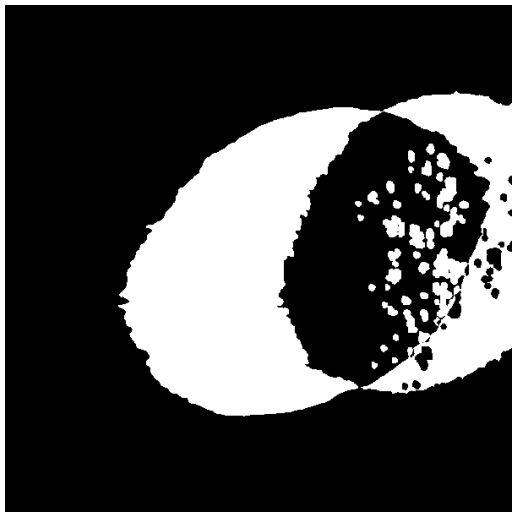
Thresholding algorithm comparison shifting



(a) Proposed segmentation.



(b) FPGA segmentation.



(c) Sample of differential segmentation.

Figure: Sample of differential segmentation from shifted frame.

Segmentation

Table: Model segmentation accuracy results.

Model	Samples	% Range	Accuracy
M0	145	73.61% - 96.46%	84.54%
M1	145	80.01% - 99.02%	96.22%
M2	155	87.77% - 99.18%	**97.10**%
M3	153	88.22% - 99.09%	96.64%
M4	151	70.69% - 98.76%	87.36%
M5	138	64.81% - 98.71%	94.14%

Model resource usage

Table: FPGA models resources.

Model	Used Slices	Usel LUTs
M0	5987 (3%)	6012 (6%)
M1	**6110** (3%)	**6111** (6%)
M2	5987 (3%)	5899 (6%)
M3	6110 (3%)	5959 (6%)
M4	5810 (3%)	5377 (6%)
M5	5988 (3%)	5588 (6%)

Conclusions and future work

- Even though results from our experiments show that the Otsu thresholding algorithm gives slightly better results than our proposed methodology, this algorithm requires to store in memory a complete frame and it performs pixel neighborhood floating point computations that can be substantially more complex to implement in hardware. In order to develop an FPGA version of the Otsu algorithm, a physical RAM must be used to be able to save the frame, and it would be very expensive in resources. For this reason, a decision-tree-based pixel classifier implemented using an array of comparators was proposed in this work, since its architecture resources are less demanding, it is more simple and it can be run in real-time.
- Feature relevance analysis is a key piece in this work, since it allows us to define the model with the best performance, where the Gray and Green channels were the best features and $M2$ resulted as the model with best performance.
infrared sensors.

Conclusions and future work

- The proposed decision-tree model, including all support modules, uses very few resources, and we are working to process another classification line at the same time using only one device.
- In this work is not discussed how to detect defects in oranges. This will be addressed in a future work using infrared sensors.
- The proposed architecture is the first stage of a classification system that will be able to classify oranges at 60 fps. This architecture uses I^2C to communicate with a microcontroller that will be able to control the I/O of the whole system. Results show that the system will be able to classify 5 ops by color and size. This represents 1 fruit-ton of classified fruits in approximately 18 minutes.

Acknowledgment

This work was supported by CONACyT, Mexico, under research grant No. 933607.

References