Abstract: The field of food industry is nowadays governed by some of the toughest regulations ever seen in industry in order to maintain the quality of practices such as product traceability and hygiene. Beside these it is required to maintain flexible processes in order to allow alternative products specifications, while maximizing robotic utilization. It is therefore no surprise that this field is very active in introducing automated systems to perform many tasks. The recent advances in semiconductor technology allow to design and implement advanced mathematical methods on low cost hardware platforms. It is a desiderate for smart sensor systems to implement mechanisms similar to the humans mental activity of combining information for the goal of recognizing a measurand that is hardly distinguishable with a single sensor. This presentation will introduce the concept of virtual sensor. A vision based contactless weighting sensor will be presented together with the mathematical foundations required for building and calibrating it. Its real-time performance is demonstrated through the implementation of an automatic sorting system. Further improvements and optimization techniques are introduced in order to make the system able to maximize yield of edible product, reduce labor requirements of the complete system, to provide a process compatible with most current plants and to improve overall plant productivity.

Key–Words: sensor systems, virtual sensor, approximation methods, vision, food industry

1 Introduction

The chapter Vision Based Sensor Systems in Food Industry [5] addresses the problem of a system based on multiple vision sensors in estimating the mass of an object. It is a desiderate for smart sensor systems to implement mechanisms similar to the humans’ mental activity of combining information for the goal of recognizing a measurand that is hardly distinguishable with a single sensor. The field of food industry is very active in introducing automated systems to perform many tasks. Among these, the sorting of food or biological material is very important. An original weighting sensor was developed. Its real-time performance was demonstrated through the implementation of an automatic sorting system.

The introduction of computer vision based equipments in food processing industry is targeting to achieve the following goals [2]:

- To maximise yield of edible product;
- To reduce labour requirements of the complete system;
- To provide a process compatible with most current plants;
- To improve overall plant productivity;
- To maintain flexible processes in order to allow alternative product specifications, while maximising robotic utilisation;
- To maintain the quality of practices such as product traceability and hygiene.

A basic architecture is presented in Figure 1 [3]. Vision sensor concept denotes vision systems for color, shape and size measurements. Regulators and sensors are seen as the interface to control the production process: controlling and measuring "traditional" parameters like temperature, humidity, speed of conveyor belt, etc.

Main processing unit is the decision maker in the on-line implementation of the system. It is continuously updated from the vision sensors and the other regulators/sensors connected to the system. If the MPU receives a warning from the vision sensors, it will analyze the problem from its knowledge of current sensor settings for a precise diagnosis, and if needed change the settings of the relevant regulators.
2 A low cost architecture for measurement and control

Only one decade ago, designing and implementing an image acquisition system for a real-time application required high-tech hardware and software solutions. These solutions were expensive and not widely available.

The main drawback was that implementing a real-time acquisition system required the use of an expensive acquisition board, since the CPU power at that time barely covered the processing power required by such an application.

But the use of an acquisition board often meant that the application designer was forced to use a compatible camera for that board. The higher the demanded quality of the captured images, the more expensive was the final solution.

In this chapter a USB based low-cost image acquisition system will be introduced together with its implementation details related both to the hardware and software implementation. In this way the cost of the final product can be lowered.

The USB cameras open new implementation possibilities because, to nowadays PC, it is possible to connect several cameras. In this way, with appropriate operating system and device drivers it is rather simple to implement a multiple camera system.

There are two ways to support multiple USB cameras connected to the same PC. One is to make use of multiple USB cameras each with its own device driver. Then the application’s software must provide a mean to control the devices, especially from the resource usage point of view. Also, the device driver must allow the OS to use all the connected cameras at the same time.

Another way to support multiple USB cameras is to use a device driver that can handle multiple cameras. In this way the control over the resources is made inside the device driver and the application’s software does not need to deal with the control of cameras resources.

A real-time application that uses an image acquisition system has, generally, the following main modules:

- the acquisition module: which is responsible for interfacing the application with the image acquisition system
- the processing module: which is responsible for applying application specific operations on the acquired images.
- the control module: which is responsible for issuing application specific commands, using the results from the previous module.

An important aspect of the real-time application is that the operations from the different component modules must be synchronized and made in real-time (that means that data buffering is acceptable only for a well defined and application specific time). Having these constraints, the interface between the acquisition and the processing module must be very carefully designed.

The first problem that appears is the synchronization between the two modules. There are two time intervals that are of interest: the first one is the time difference between two successive frames (the inverse of the frame rate), and the second one is the time needed for the processing module to complete its task. In order to keep the modules synchronized, some restrictions must be applied. In order to not have unprocessed frames, a restriction must be imposed on the processing time, and that is that the processing time must be less than the time interval between two successive frames. This restriction can be relaxed only by increasing the time interval between frames, that means either lowering the frame rate or by allowing that part of the frames are dropped, only each n-th frame being processed.

Using this approach, the design of the application requires multiple capture threads, one for each camera, and one thread for processing. The capture threads are running free, the acquisition data is stored in a thread safe manner in an input buffer, and the processing thread reads the data stored from that buffer. It is worth mentioning the following problem encountered with this approach: if the cameras are not synchronized, then the captured frames from the multiple connected cameras may not have the same time.
stamp. Because of this, the information from the cameras cannot be used to accurately describe the same scene.

For an application that requires the description of a scene using multiple cameras, a more suitable approach is to synchronize the acquisition module with the processing module. This can be done by controlling the acquisition. In this case a separate control thread is required, thread that will be responsible for controlling both the processing thread and the capture threads, and in this way the synchronization, both between the acquisition module and the processing module, and inside the acquisition module (between the cameras), can be achieved.

![Timer](image1.png)

**Figure 2: Threads setup**

![Low cost hardware architecture for monitoring and control](image2.png)

**Figure 3: Low cost hardware architecture for monitoring and control**

The controlling thread (see Figure 2) commands when the capture should begin and, when the capture completes (the captured frames from all the cameras are ready for processing), triggers the start of the processing thread. After the processing is completed a new capture command can be issued to the cameras. In this way, the frame rate at which the images are acquired is given by the time needed to process the captured frames.

A low cost hardware architecture that can accommodate for the software setup described previously and offers in the same time possibilities to interact with the process in terms of measurement and control is presented in Figure 3.

### 3 Mathematical methods for multiple sensor systems

The case to which these regression techniques were applied is that of a vision based sensor for measuring the mass of biological material. The sensor is intended to be interfaced to a sorting system. A conveyor belt carries the objects that have to be sorted, imposing real-time conditions on the measurement and control system. The approach used in this work is based on several ideas:

- The vision system is seen as a collection of *virtual sensors*, each of these being capable of sensing a certain geometrical feature of the objects in the scene.

- A Principal Component Analysis is made upon the series of measurements obtained by inspecting a training set of objects.

- The measurands (features) which demonstrate the best correlation with mass as well as the larger variation with mass, are considered to be the measurands of a multiple input sensor whose output is the mass.

- A multiple regression algorithm is used to locally compute the transfer surface of the sensor.

### 4 Experiments and results

A test system was implemented according to the architecture of Figure 3. Three applications were implemented on the PC.

The first one allows the acquisition of the images containing the objects of interest and off-line processing of these images in order to output a set of features quantified. Figure 4 shows in the user interface the images acquired with two cameras, one oriented from the top of the scene (left image) and other laterally oriented (right image).
The basic operations implemented in this off-line application are:

- binarization,
- labeling,
- filtering and detection of contour and main axis,
- extraction (computing) of those object parameters which quantify the features.

![Table 1: Principal component transformation matrix](image)

![Table 2: Fractional variance explained by each component](image)

From the measurement series provided by this application, a Matlab implementation of the Principal Component Analysis was used to find the two factors upon which a regression can be made to estimate the object mass. The features used in the analysis were the areas of the object, the estimated volume, the angles of the main axes, the moments of inertia relative to these axes, the extents and the shape factors. The principal components presented in Table 1 were selected as those with maximum values in Table 2. After that, Table 3 was used in order to identify which of the initial variables better explain the mass. As it can be seen, these factors were the object area in the left image and the estimated volume.

![Figure 5: Application for calibrating the weight sensor](image)

A second application (see Figure 5) was developed for constructing the sensor transfer surface as shown in Figure 6. The parameter $R$ of this application controls the influence of the measurement in interpolating values on a grid defined in the space of measurands.

Note that the transfer surface constructed upon a training set of known objects exhibits spikes before the removal of the outliers.
A third application was developed for real-time control of a selecting machine. The machine must select objects on a conveyor belt into 6 categories according to their mass. Figure 7 shows the diagram of this application.

At the start of the application two files are loaded. The first one contains the parameters of different modules. The second file contains the “lookup table” that will be used to estimate the weight of the snails. This file is an adapted GRD file, customized to accommodate the specific needs of this application. The content of this file is used to build a uniform rectangular grid, in the space defined by the area of the object and the estimated volume of the object. The grid associates to each point defined by the pair (area, volume) the interpolated weight. This grid was built using a training set of snails of different category. Each snail was weighted, and then exposed to the two cameras in 4 different orientations. The resulting images were then processed off-line using the same routines that will be used for the online processing. Some additional parameters were computed for use by PCA. The results obtained for all the snails from the training set, more precisely the measured area of the object of interest, (captured by the top camera) and the estimated volume (computed based on the object from both images) are then used in conjunction with the corresponding measured weight to build an interpolant. Using this interpolant, and the extreme (min, max) values of the area and volume of the training set of snails, the uniform, rectangular grid is built.

The application starts by generating two threads, one for each camera. Next, a third thread is started, thread that will be responsible for controlling the actuators. This thread will execute the commands from a queue which will be updated after the processing of a new pair of frames is completed.

The main thread is the processing thread. It is responsible for acquiring, processing and updating of the commands for the actuators. It has two timers attached to it: one is the capturing timer, which will raise a capture event which will be passed on to the camera threads. This is a “slow” timer, used to assure that the captured frames are equidistant and that the capturing does not interfere with the processing of the already captured frames. Also, having a constant capture rate, it allows for a better estimation of the position of the objects in the image relative to the actuators.

The second timer is a “fast” timer, and it has the main purpose to start the processing of the captured frames as soon as possible. The rule here is that it “scan” the status of the capturing threads as fast as possible and when both threads report that the capturing of the frame is complete it initiates the processing of the acquired frames. The “low” timer triggers the capture of a new frame only if the processing of the current frame pair was completed. Otherwise, if the processing of the current frame pair takes longer that the capturing period, frames will be dropped and, possibly, we will miss some scenes.

Assuming that the processing timer has expired and that both frames are ready, the frames are passed to the application processing module that filters the objects.

The first task is to see if there is data coming on the serial port connected to the speed sensor of the conveyor belt. This information will be used to compute the timeout for the corresponding actuator of the sorter.

Next, a flag is marked so that the application
is placed in the Processing mode. The last frames are taken from the circular buffers were the camera threads place their data. The frames are then passed to an image processing module that will label the objects in the images, filter the objects with dimensions below a certain threshold and then extract various features of the remaining objects. Although the ShapeFactor and the Inertia are not used for weight estimation, these will be used for further filtering of the objects, for example to avoid processing objects that are too close to each other, case in which the estimation error will be too large to allow correct sorting.

After having all the required information for all the objects in the two frames (one from top camera and one from side camera), the next task is to make the correspondence between the two frames. That is, matching the objects in the top view with the objects in the side view. Because of the fact that in the side image the image is distorted (the perspective effect) a mapping between the two images was previously built. After this is done, the volume is estimated using the Cavalieri principle.

Having the area of the projection in the top frame and the estimated volume, we can now try to estimate the weight of the object so that we can determine its corresponding category and in this way “compose” the appropriate command for the actuators. The weight is taken from the “coordinates” given by the area and the volume in the table of interpolated weight data.

The application was developed in Visual C++.

5 Conclusion

An original weighting sensor was developed. Its real-time performance was demonstrated through the implementation of an automatic sorting system. The features used in the analysis were the areas of the object, the estimated volume, the angles of the main axes, the moments of inertia relative to these axes, the extents and the shape factors. The principal components were selected in order to identify which of the initial variables are explaining better the mass.

A PC application was developed for constructing the sensor transfer surface. Finally, an original automatic sorting system based on vision sensors was implemented and optimized.

References:


