J Supercomput (2019) 75:1026–1037
https://doi.org/10.1007/s11227-018-2284-y

Efficient image-based analysis of fruit surfaces using CCD cameras and smartphones

J. A. Álvarez-Bermejo[1] · **D. P. Morales-Santos**[2] · **E. Castillo-Morales**[2] · **L. Parrilla**[2] · **J. A. López-Ramos**[3]

Published online: 1 March 2018

Abstract Today's smartphones make a broad variety of sensors (gyroscope, magnetometer, camera, accelerometer, GPS, etc.) readily available and easily accessible through different APIs, favouring the acquisition of data. An everyday usage is the measurement of physical parameters, like sound or acceleration. The advances in terms of level of integration and its application to embedded devices power consumption and wide adoption of system on chips and more recently multiprocessors system on chip mean that a new sort of applications can be addressed. Applications are backed with powerful computing devices depending on batteries. Using these resource-limited devices and their parallel power efficiently is a challenging task. To fully exploit the potential of these hardware devices, parallelism has to be carefully applied to the most resource demanding parts of the application. This paper proposes an efficient image composition method to analyze fruit surfaces using CCD cameras and smartphones. The image composition is done by capturing video from which redundant frames are

✉ J. A. Álvarez-Bermejo
jaberme@ual.es

D. P. Morales-Santos
diegopm@ugr.es

E. Castillo-Morales
ecastillo@ugr.es

L. Parrilla
lparrilla@ugr.es

J. A. López-Ramos
jlopez@ual.es

1 Department Informatics, Universidad de Almería, Almería, Spain

2 Department Electronics, Universidad de Granada, Granada, Spain

3 Department Mathematics, Universidad de Almería, Almería, Spain

disposed using a data-parallel local feature detector. Relevant frames are then stitched using direct methods. The proposal was tested in the case of calculating the damaged surface of cherries.

Keywords Image processing · Homography · Direct methods

1 Introduction

The importance of a post-harvest system is defined in a FAO document [1] as encompassing the delivery of a crop from the time and place of harvest to the time and place of consumption. It is a complex aggregate of logically interconnected functions or operations within a sphere of activity. Being able to gather augmented information on the food that is offered to the customer is not only a value for the food chain but for the health. The term chain or pipeline highlights the functional succession of various operations but tends to ignore their complex interaction. Any inconvenient affecting the system (as damages in the production, etc) is translated directly into loses. In addition, finding an objective method by which the production can be evaluated is a difficult task; experts visually evaluate the quality of the production. Using image processing techniques is a solution but there is an extra condition; the system should be portable and mobile so that the product tracing can be done fast and at anyplace.

Consumers associate directly injuries or defects on the surface of the fruit with the freshness or maturity of food products. A practical application where the inspection of the surface is also needed is the classification of the product according to the damages shown in the surface. This classification is directly related to the target market. This sorting focuses on classifying by percentage of damaged surface. A way to automate this process is to acquire images of the fruit and analyze its features using image processing techniques. Automated estimation of the damaged surface by means of image processing techniques has a direct impact in accuracy, objectivity and repeatability. However, when dealing with color images, one of the biggest difficulties is the RGB space coordinates (native color space for most image acquisition devices), and it is device dependent. It is common to convert the default color space to other more convenient, for measuring or representing the color of fruits [2].

The image processing methods have been adopted by the agriculture community, but related to this there are always a number of aspects to consider apriori, when applying these techniques, like the image acquisition, which is usually designed to be run on a desktop computer, which is clearly not practical. An alternative is the implementation of the acquisition system using a MPSoC-like device, that is, a smartphone. Actually, a smartphone is a non-expensive portable computer with a very high processing power. In addition, the myriad of built-in high-resolution sensors and CCD cameras provided with these devices makes them the ideal solutions for many tasks in the agriculture and farming scenarios. The capability to acquire and process images allows to obtain objective and accurate information on the tasks that have traditionally been based on experienced workers, see the paper [3], where authors gathered data through the smartphone's built-in camera from bananas to report their ripeness just by measuring the color. For the objectives of this paper, a number of issues need to be solved; among them it is how to deal with the color space to process the surface injuries of the fruits in light-independent scenario, how to gather the information of the surface (images) to efficiently unfold the surfaces and the segmentation processing method selected to quickly identify the regions of interest.

Related to the first issue, the image gathering method in [4] a system using binocular cameras is used to capture images from different angles (each camera gathers non-overlapping images and analyzes the images separately); this way authors avoid to create a computational model; as we did in this paper, where the movement of the camera is described, this proposal implements an advance to the one raised in [5] where authors statically collect a chunk of images that were analyzed independently. Both approaches were acceptable to process the information about fruit surfaces, but the main drawback is the lack of portability and the time needed to obtain the results. The work presented in this paper in addition allows to analyze non overlapped images but with the help of homographies that collect all the visual information regarding the surface as it was unfolded. Related to the color scheme, a profound debate is opened by researchers when choosing a color space to operate in, another aspect to consider, opinions diverge. The HSV color scheme suits well the processing needs targeted in this paper as it represents the true color and allows a better understanding of the features. Our proposal builds the surface by composing from the different frames selected from the video stream; when ready it is moved to the HSV space to make it light independent as it separates luma (image intensity) from chroma (the color information). And finally, with respect to the segmentation, algorithms are well known, and in this paper the Canny algorithm was used.

System on chip [8] (SoCs in advance) and their recent advances (MPSoCs) are the reason of the existence of current mobile technology. The devices (mobile systems such as smartphones are a fundamental extension to our digital lives due to their rich variety of features) are built on complex hardware and software platforms. From the hardware side, the MPSoC is a solution to meet the computational requirements of the society, and these devices are pushed to improve processing power against battery restrictions applied to computational power. Examples of current examples of MPSoCs typically integrate ARM multiprocessors with GPUs. A hyperconnected technology, not only to the Internet but to a wide variety of sensors that are continuously adding valuable information (i.e., smartphone sensors provide us with the information of the rotation and angle to feed the homography model used in this paper). As a consequence, more processing power is being put on these mobile processors that are now multicore architectures linked to low-power GPU devices designed for smartphones. Moreover, MPSoCs even incorporate coprocessors like DSPs; for example, the Hexagon DSP in Qualcomm SoCs includes a DSP–SDK. From the software perspective, these devices are widely accepted to be under the operation of the Android OS, which is its main advantage as well as its main drawback. Android is powered by a Java Virtual Machine, forcing us to search for native when requiring high performance results. To overcome this, Android offer APIs to develop sections of code designed to boost application sections that are computationally high demanding. Several programming paradigms are available for achieving this: OpenCV, BoofCV and RenderScript among others. Developing a real efficient code for such architectures is a complex task [9]. These architectures can be considered as the conventional UMA CPU/GPU devices, with memory being a sort of high-bandwidth communication channel.

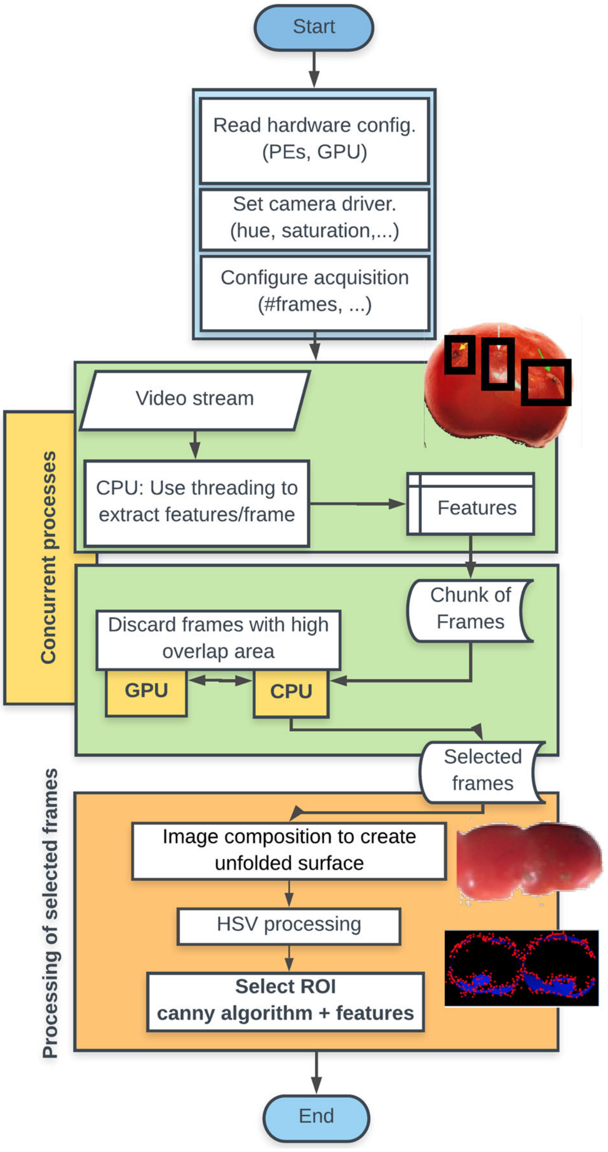
When considering both sides, the image processing issues raised and the MPSoCs platforms where the solution is to be developed and deployed, real-time image analysis is concerned. Normally, these problems are based on detecting features of the images fed to the processor [10] by the CCD camera sensors. These features are translated to pixels of the image known as keypoints [11]. Once located, they need to be tagged for later operations [12]. One of the most popular algorithms is speeded up robust features (SURF), which is a feature detector, scale and rotation invariant [13]. Current smartphones deliver high-resolution images; therefore, computational power is the corner stone. Algorithms must meet strict resource limitations and almost reach real time or at least provide a solution in an acceptable time. Parallelism is the solution to reduce the response time and algorithm efficiency.

In any of the definitions, what this paper pretend is to provide a fast method to calculate the impact of the damage that is affecting the fruit surface and detect the potential food loss due to an inappropriate transportation, manufacturing, storage, etc. The image processing applied is based on that the macroscopic changes that appear are used to assess the effect of damage. The objective of the work is to develop an application designed for Android Smartphones to objectivity the measurements using machine vision [7].

2 Materials and methods

Unfolding of the fruit surface consists in constructing a 2D image containing the contour of a 3D element. It is an interesting procedure to analyze the features of the surface of the 3D element. The technique implemented, as shown in Fig. 1, to unfold and analyze the surface of each fruit is the image stitching [14] and can be automated through direct or feature-based methods. Feature-based methods match image features, whereas direct methods (faster than feature methods [15,16]) use all image data and minimize the pixel-to-pixel dissimilarities. Figure 1 shows the procedure of the implementation; in a first stage the hardware is configured and set. After this, the video starts to capture frames (video stream), multithreading is used to detect salient features for each frame. These features are stored in internal data structures. Concurrently with this, when a set of frames is stored and their features are detected, an area overlap optimization is started to detect which frames will be used to create the stitch. Discarded frames are those whose surface overlaps with the first frame of the set under study. Once the frames are selected, they are sent to the stitching process.

To contextualize the implementation, the supporting theories and methods (defined and described in [15,16]) used as reference are described next. The image stitch (or projective model) operates on homogeneous coordinates $\tilde{x}$ and $\tilde{x}'$, $\tilde{x}' \sim \tilde{H}\tilde{x}$—just consider (as seen in Fig. 2) that a certain point (with 3D coordinates) located in the surface of the fruit is now translated into 2D coordinates in each of the frames (where such point appears), where $\tilde{H}$ is a 3×3 arbitrary homogeneous matrix. The resulting $\tilde{x}'$ coordinate should be normalized to get a non-homogeneous result x' (with the

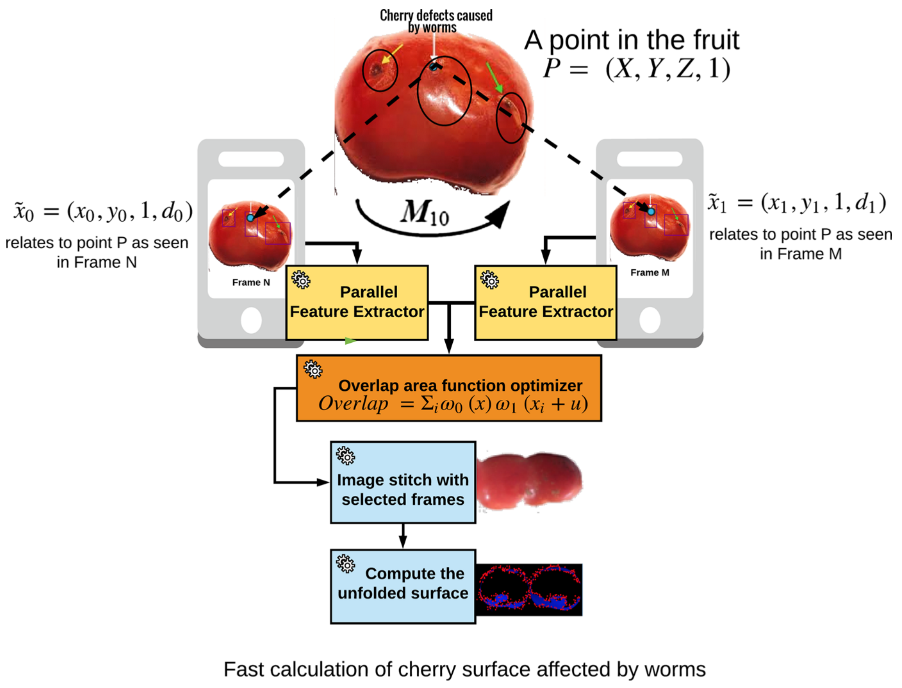


Start
Read hardware config. (PEs, GPU)
Set camera driver. (hue, saturation,...)
Configure acquisition (#frames, ...)
Video stream
CPU: Use threading to extract features/frame
Features
Concurrent processes
Chunk of Frames
Discard frames with high overlap area
GPU
CPU
Selected frames
Processing of selected frames
Image composition to create unfolded surface
HSV processing
Select ROI canny algorithm + features
End

Fig. 1 Description of the implementation

consideration that an alternative normalized device coordinate [17] was implemented, where pixel coordinates varied from (− 1, 1) in longer axis and from (− a, a) in the shorter, a is defined as inverse of the aspect ratio, and therefore, for an image sized with width W and height H, the equations mapping $\bar{x} = (\bar{x}, \bar{y})$ to $x = (x, y)$ are $x = \frac{2\bar{x}-W}{max(W,H)}$ and $y = \frac{2\bar{y}-H}{max(W,H)}$), see Fig. 2.

$$x' = \frac{(h_{00}x + h_{01}y + h_{02})}{(h_{20}x + h_{21}y + h_{22})} \quad \text{and} \quad y' = \frac{(h_{10}x + h_{11}y + h_{12})}{(h_{20}x + h_{21}y + h_{22})} \tag{1}$$



Cherry defects caused by worms
A point in the fruit
$P = (X, Y, Z, 1)$
$\tilde{x}_0 = (x_0, y_0, 1, d_0)$
relates to point P as seen in Frame N
M_{10}
$\tilde{x}_1 = (x_1, y_1, 1, d_1)$
relates to point P as seen in Frame M
Frame N
Frame M
Parallel Feature Extractor
Parallel Feature Extractor
Overlap area function optimizer
$Overlap = \Sigma_i \omega_0(x)\omega_1(x_i + u)$
Image stitch with selected frames
Compute the unfolded surface
Fast calculation of cherry surface affected by worms

Fig. 2 Description of the app model focusing on the accelerated image composition method

With respect to [16] the perspective projection is a permutation matrix that permutes the last two elements of homogeneous 4-vector $p = (X, Y, Z, 1)$. As depth values cannot be sensed (due to the nature of the cameras used—Samsung Galaxy S5—K is created using f as the focal distance, providing high-quality results in stitching images), the z-buffer is ignored; thus, K is the intrinsic calibration matrix.

$$\tilde{x} \sim \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} p = [K \mid 0]\, p \tag{2}$$

Any 3D point p is mapped to an image coordinate $\tilde{x}_0$ in the position 0 of the camera through the combination of E_0 (Euclidean motion) and P_0 (perspective projection):

$$x_0 = \begin{bmatrix} R_0 & t_0 \\ 0^r & 1 \end{bmatrix} p = E_0 p \quad \tilde{x}_0 \sim P_0 E_0 p \tag{3}$$

Planar scenes are considered in this paper (as stated above no information on depth coordinates of pixels is available) by adding to Eq. 2 a general plane equation $\hat{n_0} \cdot p + c_0$. The mapping equation is therefore reduced to $\tilde{x_1} \sim \tilde{H}_{10}\tilde{x}_{10}$ being $\tilde{H}_{10}$ is a general 3×3 homography matrix and $\tilde{x}_0$ and $\tilde{x}_1$ are 2D homogeneous coordinates now. Once defined both the coordinates and motion models, a metric to instrument error and a search method are needed to define the match between a pair of images (see Fig. 2).

And a method to accelerate this process is needed (see Sect. 2.1). Given an image as reference $I_0(x)$, the task consists in finding where in $I_1(x)$ are located the pixels of interest. $I_1(x)$ is selected as a potential frame for the stitch if the similar pixels are minimum. When processing the pixels similarities, it should be considered that a subset of the pixels may fall outside the original image boundaries. (These pixels should be discarded.) A weighted minimum to the sum of the squared differences function is proposed (where $u = (u, v)$ is the displacement and $e_i = I_1(x_i + u) - I_0(x_i))$ the residual error).

$$\text{Calculated error} = \sum_i w_0(x) w_1(x_i + u) \left[I_1(x_i + u) - I_0(x_i)) \right]^2 \tag{4}$$

Equation 4 is the foundation for the function to minimize, which is the overlap area computed as

$$\text{Overlapping area} = \sum_i w_0(x) w_1(x_i + u) \tag{5}$$

Only the selected frames satisfying the condition Overlapping area $<$ Threshold are computed. The image composition using the image stitching was implemented using OpenCV [18], BoofCV [19] and a particular version where we used SURF [13] to implement keypoints, descriptors and matching; for this case, the stitch was implemented by stitching images reducing the pixels dissimilarities (direct methods), see Sect. 2.1. After creating the images, the analysis of the surfaces is conducted in the sequential version, and in the parallel version, in the sequential version the processing is assigned to one specific core. The parallel version was implemented using RenderScript. To detect the region of interest, the Canny algorithm [20,21] was used.

2.1 Parallel feature extraction to discard overlapped frames

As sketched in Fig. 2, there is a need to implement a feature detector able to compute the level of similarity (repetition rate) between two images. We have considered that *redundant* images can be discarded from the homography. To this end the SURF method [13] was used to perform feature extraction and providing local correspondence for a given pair of images. The main interest of this approach lies in its fast computation of operators using box filters, thus enabling real-time applications. The proposed implementation of the SURF algorithm is written using different versions; one is adapted to use with OpenCV, the second one uses the stitch implemented in BoofCV, and the third version, a *native* approach, was written in C++ and implements the stitch using [13] together with direct methods.

The key concept of the SURF approach is the *integral image*, with its image convolution that is faster. It is a construction to efficiently gather sum values form a pixel grid $\Omega = [0, N-1] \times [0, M-1]$. Let p be the digital image defined over Ω, then the integral image of p for $(x, y) \in \Omega$ is:

$$I(x, y) := \sum_{0<i<x} \sum_{0<j<x} p(i, j) \tag{6}$$

The convolution of the image p with a 2D uniform function B_Γ (**B** box filter) over $\Gamma \in \Omega$ as $B_\Gamma(x, y) := 1_\Gamma(x, y) = \begin{cases} 1 \; if(x, y) \in \Gamma \\ 0 \quad \text{otherwise} \end{cases}$ whenever the domain Γ is rectangular and therefore separable in rows and columns coordinates, image p can be expressed directly from the integral image I as: $\forall(x, y) \in \Omega$, $\Gamma = [a, b]\times[c, d]$ $(B_\Gamma * p)(x, y) = I(x-a, y-c) + I(x-b-1, y-d-1) - I(x-a, y-d-1) - I(x-b-1, y-c)$, this expression is used to find the sum of pixel values contained in the rectangle area Γ.

The steps of the SURF algorithm include *finding keypoints* refer to salient features of the image [22]. To detect keypoints the determinant of the Hessian matrix is used. *Keypoint descriptors*, orientation and locality descriptions are taken into account; in this stage Haar filters are used to face rotation invariation. *Matching* is used to locate common points in two images (see Fig. 2), and this paper used the Euclidean distance. Keypoints are matched between reference and actual images (see Fig. 2) when the Euclidean distance is < 0.6.

The parallel nature of the algorithm is data parallel. Different stages of the algorithm cannot be processed in parallel due to their strict dependencies. But as the image and the rectangle area Γ are separable, it is possible to create computational kernels to derive the computation to different cores. Therefore, the integral image (see equation Eq. 6) can be segmented and submitted to different processing entities. This way, the image I is calculated jointly by all the available and assigned cores. Each computing unit retains a segment of the integral image I, and the Hessian determinant (DoH) is calculated on the part of the integral image I that is sent to each core. Descriptors are calculated accordingly to the segment of the image. The matching stage consists in every core/kernel calculating the (euclidean) distance between each one of the local descriptors of the current image against the set of descriptors of the reference image. The parallelization is straightforward.

3 Results and discussion

In these experiments the SURF feature detector and descriptor were implemented in OpenCV and BoofCV. These algorithms were executed to extract the homographies of all the frames as fed from the CCD camera. A third method was tested with the parallelization of the SURF steps (keypoint detection, descriptors and matching); this third version skipped the image composition using the feature-based method. Instead, the composition was achieved using the non-discarded frames. To compute correspondences between detected features, the euclidean distance was used. It was considered that correspondence is found between descriptor d_i from image I_n and descriptor d_j from image I_m if the euclidean distance is less than 0.6. The testbed platform for the experiments was a conventional Samsung Galaxy S5 smartphone. Table 1 shows how the image composition is exposed to the SURF method (feature-based) in contrast to direct methods. The number of frames computed was 1000. The

Table 1 Time to compute the stitch without parallelism and frame discarding

Overlap (%)	Homography (s)	
	SURF	Direct
< 20	601.25	21.39
< 10	502.34	9.54
< 2	366.9	2.07

Table 2 Comparison of the three versions during the stitch composition

Frames	Method	Mean (ms)	Median (ms)	SD (ms)
0–500	OpenCV	641,11	532,71	363,44
500–1000	OpenCV	440,91	372,20	368,21
0–500	BoofCV	672,27	551,27	400,21
500–1000	BoofCV	481,81	402,02	391,12
0–500	Surf + direct	602,38	505,83	311,24
500–1000	Surf + direct	420,37	399,12	302,74

feature-based methods are preferred, but the direct methods are faster. As seen, Table 1 shows the extremely long times taken by feature-based methods; it is also shown that as the percentage of overlap is reduced also the time needed to prepare the stitch is significantly reduced.

The SURF method in Table 1 is not parallelized. Therefore, it is the aspect that suffers most the impact of the overlapping factor. Table 2 shows the execution of the steps described in Fig. 1 until the homography is created. Figure 2 shows that for the third method the frames extracted from the video are analyzed and compared with the first (reference) frame.

Once the overlapping conditions are satisfied, the first (reference) frame and the frame satisfying the overlapping condition are marked to create the stitch. From this moment, the reference frame is now the selected frame. And the process is repeated until the frames buffer is emptied. Tests were run on a set of 1000 frames and just for the $< 2\%$ case. In Table 2 the three methods are contrasted: the feature-based image stitching method parallelized using OpenCV, also parallelized using BoofCV library and finally the combination of the SURF (feature-based method) together with the direct stitching but using exclusively the selected frames; this way the direct method operates on a *feature-based* basis. The camera was moving in rotation to capture the contour of each element under study. As results show, the first 500 frames needed more computation (keypoints detection, descriptor and matching) as the background was not the same as in remaining frames. Our method was tested for the case of selecting only frames with 1–2% of overlap. This case can be seen in Fig. 3 where OpenCV is shown to be faster than BoofCV in every test executed. But in contrast it is interesting that the option of discarding frames using features (surf) and switching to direct methods on the set of the selected frames also shows performance gains.

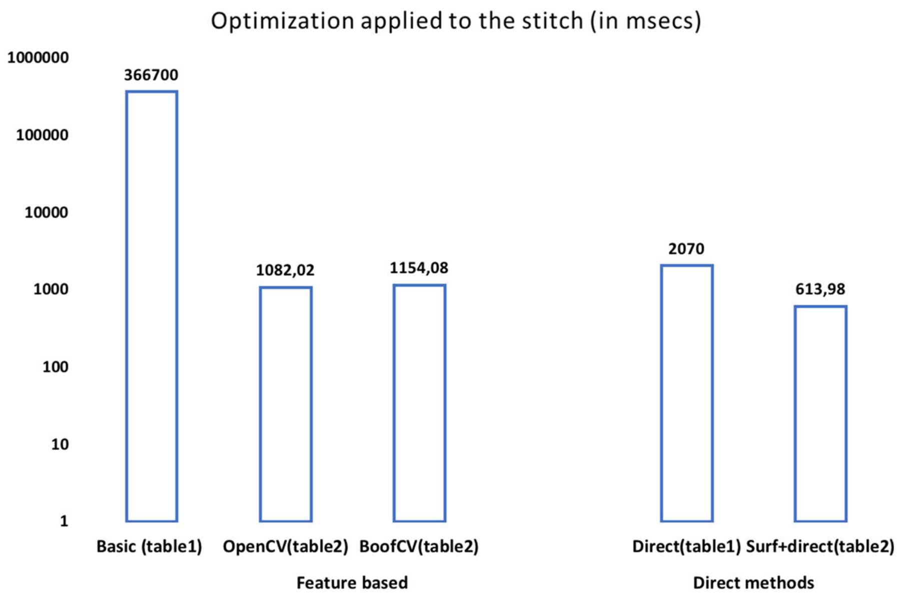


Fig. 3 Comparison of the performance achieved

Table 3 Elapsed time in computing the surface on the stitched image (see Fig. 2: compute the unfolded surface)

Overlap (%)	Analysis of the unfolded surface (ms)		
	Sequential	Threaded	RenderScript
< 20	2315.03	1793.94	931.12
< 10	1996.75	1129.45	765.23
< 2	1827.31	1011.83	704.79

Our method is exposed to the image composition using SURF versus direct methods to create it. Once the image stitch is finished, Table 3 shows the elapsed time in milliseconds taken to analyze it. The analysis was implemented in three different ways as it is a common paradigm. Firstly, the surface is analyzed with a sequential method that processes the whole image using one core (region detection and points of interest (see Fig. 1). Table shows a version implemented using user-level threads. One thread per core was the configuration selected. And the third version included the RenderScript API to compute images using the CPU and GPU, but the results for the RenderScript version were not as favorable as it might be expected.

The relevance of the proposed method is seen when unfolding surfaces to compute the surface of the fruit. The device camera feeds the video to the processor. Frames with relevant information (satisfying the overlap area condition) are selected. From the selected frames, the unfolded surface is constructed. This image is analyzed to detect defects.

4 Conclusions

A method to evaluate the quality and freshness of the products, using image processing, has been evaluated using smartphones to analyze the external conditions of the fruit. It has been an aspect to focus on the optimization of the processing time and the power supply requirements. The results can affect not only the use/management of the specific lot of product, but the general procedures in order to obtain better tracing methods.

The image composition was a key part of the proposed solution. The overall process consists in several differentiated parts that have been optimized. In a first instance, as the video is produced, the frames are processed individually, extracting their salient features. The frames are internally batched in chunks to remove redundant (excessively overlapped frames). These frames are then sent to the image composition stage. When the frames' buffer is emptied, then the composition is created and the analysis stage is started. The analysis stage was studied in its sequential nature and through two other performance oriented versions, a threaded-based version and a version where CPU and GPU (if present) are used.

We can conclude that it is possible to conduct these tests using conventional embedded devices and commercial operating systems like Android with a reasonable accuracy level.

References

1. FAO UN (2017) Post-harvest system and food losses. Retrieved from http://www.fao.org/docrep/004/ac301e/AC301e03.htm. Accessed May 2017
2. Lang C, Hübert T (2012) A colour ripeness indicator for apples. Food Bioprocess Technol 5:3244–3249
3. Intaravanne Y, Sumriddetchkajorn S, Nukeaw J (2012) Cell phone-based two dimensional spectral analysis for banana ripeness estimation. Sens Actuators B Chem 168:390–394
4. Xiao BX, Wang CY, Guo XY, Wu S (2014) Image acquisition system for agricultural contextaware computing. Int J Agric Biol Eng 7(4):75–80
5. Nagle M, Intani K, Romano G, Mahayothee B, Sardsud V, Müller J (2016) Determination of surface color of 'all yellow' mango cultivars using computer vision. Int J Agric Biol Eng 9(1):42–50
6. Zhang CL, Zhang SW, Yang JC, Shi YC, Chen J (2012) Apple leaf disease identification using genetic algorithm and correlation based feature selection method. Int J Agric Biol Eng 10(2):74–83
7. Wang L, Tian X, Li A, Li H (2014) Machine vision applications in agricultural food logistics. In: Proceedings—2013 6th International Conference on Business Intelligence and Financial Engineering, BIFE 2013, art. no. 6961105, pp 125–129
8. Pasricha S, Dutt NI (2008) On-chip communication architectures: system on chip interconnect. Elsevier/Morgan Kaufmann Publishers, Amsterdam
9. Reid AD, Flautner K, Grimley-Evans E, Lin Y (2008) SoC-C: efficient programming abstractions for heterogeneous multicore systems on chip. In: Altman ER (ed) Proceedings of the 2008 International Conference on Compilers, Architecture and Synthesis for Embedded Systems, CASES'08. ACM, Atlanta, pp 95–104
10. Demaag K, Oliver A, Oostendopr N, Scott K (2012) Practical computer vision with SimpleCV: the simple way to make technology see. O'Reilly Media, Sebastopol
11. Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (SURF). Comput Vis Image Underst 110(3):346–359
12. Mikolajczyk K, Schmid C (2001) Indexing based on scale invariant interest points. In: Proceedings Eighth IEEE International Conference on Computer Vision ICCV 2001, vol 1. Vancouver, Canada, pp 525–531

13. Oyallon E, Rabin J (2013) An analysis and implementation of the surf method, and its comparison to SIFT. IPOL J Image Process On Line, ISSN 2105-1232, preprint February
14. Brown M, Lowe DG (2007) Automatic panoramic image stitching using invariant features. Int J Comput Vis 74(1):59–73
15. Szeliski R (2006) Image Alignment and Stitching. Technical Report MSR-TR-2004-92. Microsoft Research
16. Szeliski R (2006) Image alignment and stitching: a tutorial. Found Trends Comput Gr Vis 2(1):1–109
17. OpenGL-ARB (1997) OpenGL reference manual: the official reference document to OpenGL, Version 1.1. 2nd edn. Addison-Wesley, Reading
18. Pulli K, Baksheev A, Kornyakov K, Eruhimov V (2012) Real-time computer vision with OpenCV. Commun ACM 55(6):61
19. Abeles P (2012) Boofcv. http://boofcv.org/. Accessed Nov 2016
20. McIlhagga W (2011) The Canny edge detector revisited. Int J Comput Vis 91(3):251–261
21. Xin G, Ke C, Xiaoguang H (2012) An improved Canny edge detection algorithm for color image. In: IEEE 10th International Conference on Industrial Informatics. Beijing, pp 113–117
22. Mikolajczyk K, Schmid C (2002) An affine invariant interest point detector. In: Proceedings of the 7th European Conference on Computer Vision. Copenhagen, Denmark, pp 128–142