TECHNISCHE UNIVERSITÄT DARMSTADT

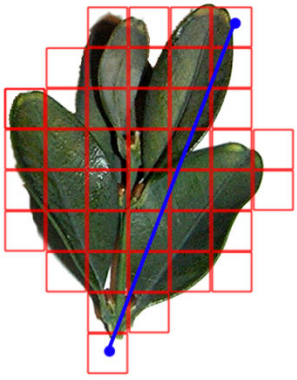
Fachbereich 20 Informatik

Fachgebiet Simulation, Systemoptimierung und Robotik
Prof. Dr. Oskar v. Stryk

Bachelor Thesis

Development and Evaluation of an object recognition for box tree cuttings

By Andreas Braun



Tutors: Dipl.-Inform. Dirk Thomas

Dipl.-Inform. Sebastian Petters

Darmstadt, April 10, 2008

1. Table of Contents

Table of Figures

2. Introduction

Visual object recognition has been an essential part of manufacturing since the very beginning of human history. The ability to check produced goods for their quality has been crucial to the progress of civilization itself as it allowed humans to improve their capabilities in production.

Starting with the industrialization there has been a need for specialists in manufactories that would check products for quality. Their main instrument is the eye, which detects differences in supposedly identical shapes with so far unmatched precision.

However the increasing productivity and output caused by improvements in machine speed and precision forced a different approach, as the human visual system wasn't able to Cope with the resulting increased requirements of detection performance.

The appearance of computers and their quickly improving speed combined with advanced sensors lead to intensive research in automated quality control systems. Today many sectors of industry rely on those to increase productivity.

This thesis is part of the BioRob joint project. Its target is to design a flexible, lightweight industrial robot inspired by the anatomy of a human arm. Instead of swivel-joints it relies on elastic wires mimicking flexor and extensor muscles. Main advantages are increased safety in work environments due to the elastic compliance of the robot arm and a better load-to-weight ratio. Consequently, this new design is suitable for utilization in new branches of industry and commerce.

One demo application is picking up and placing box tree cuttings in test tubes to evaluate a potential usage in plant nurseries. Additionally, it offers a good test for precision and flexibility due to the size and light weight of the scions.

An important part of this task is the recognition of the plants. A digital camera connected to a computer has to determine position and orientation in real-time. The following pages are describing the implementation and evaluation of an object recognition system for box tree cuttings.

3. A short history of machine vision

Beginnings

Machine vision as a discrete area of science evolved in the mid-1960s with research from laboratories and universities in the United States and Great Britain. The potential in inspection and robotic control was shown quickly. Yet a rapid adoption to practical usage was delayed for a long time due to lack of processing power and only minor automation in industry.

Starting from the 1980s, companies began to pick up machine vision systems for quality control and sorting. Renowned manufacturers of optical devices, e.g. Nikon and start-up companies emerging from university environment like Cognex(Cog08) started providing usable machine vision systems.

Image 1 - DataMan - first OCR[1] system developed by Cognex Corporation in 1982

In the following decades the amount of used vision system increased alongside with industrial robots and spread into most segments of manufacturing.

[1] Optical Character Recognition – visual detection of letters and numbers

Current State

One sector with extensive usage of computerized sorting machinery is the food processing industry. An example for a modern sorting machine is the InVision 5000 Color and Dimension System by Compac Sorting Equipment Ltd(InV08), which is able to grade several types of fruit. It uses several cameras taking images of the rotating fruit. Information is collected from up to 25 different images of a single piece. It operates with a maximal grading speed of 15 fruit per second. The user can define prototype parameters for each desired grade based on color, shape and weight. Those are determined for each piece which is sorted accordingly.

Image 2 - Sorting System InVision 5000 and graded apples in software

While this machine also handles naturally grown objects, there are several differences to the recognition of box tree cuttings. While most fruit have a desired uniform shape and can be easily compared, the plant scions have a diverse structure. Furthermore the orientation is not important for the grading and therefore not acquired.

An application of machine vision n high-tech industries is the inspection of silicon wafers for defects. Microscopes attached to a computer analyze integrated circuits using feature detection algorithms for automated calibration. Afterwards high resolution images of certain areas are compared to prototype samples in order to recognize damaged or broken connections.

A popular product for this task is the Wafer Inspection System MX61A by Olympus Surgical & Industrial America Inc(Oly08). It is capable of processing images generated by different microscopic illumination methods[2], each suited for certain applications.

Image 3 - Wafer Inspection System MX61A and a deep ultraviolet image of a CPU

A final example for a rather novel approach on finding flaws in an object or undesired materials in a probe is Heat Flux Thermography(Akt). It uses a small heat impulse to increase the probe temperature by a few tenths of a degree. As heat flux is dependent on material and volume, a sensitive infrared camera is able to recognize defects such as trapped air, fissures or corrosion below the surface. As can be seen in the picture below these areas have a distinct color in the images taken and are selected using standard algorithms for edge detection. This new method seems promising as it combines accuracy of well-known methods based on x-radiation or ultrasound while being less destructive than the first and more versatile than the latter.

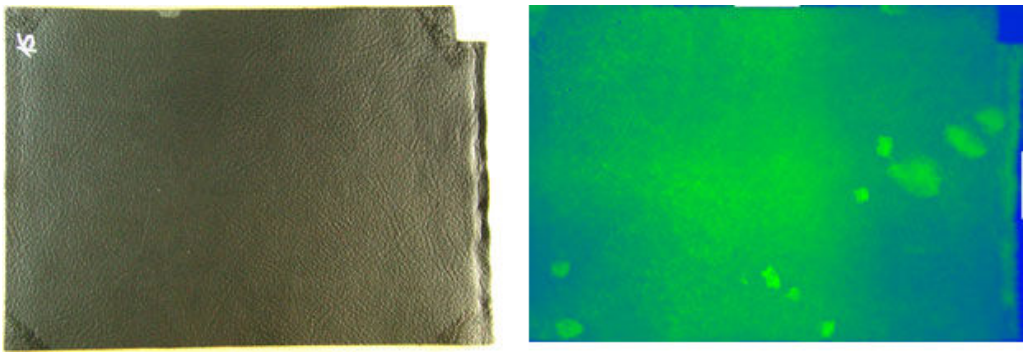


Image 4 - Photo of a leather seat cover with repaired spots on the backside made visible as green spots in the Heat Flux Thermography

[2] Bright field illumination, dark field illumination, fluorescence illumination, differential interference illumantion(Nik08)

4. Developing a test platform

Hardware specifications

The image-capturing device in this project is an IEEE1394[3] camera. The model used is a Marlin F-080C 30fps edition(All08) in combination with a Pentax TV Lens. This model was selected as it combines a high resolution sensor of 1024x768 pixels with a fast response rate of 30 frames per second.

In the project setup the camera is mounted to a fix place with position and dimensions of the conveyor belt known. Consequently image coordinates can be easily converted to world and robot coordinates. Image transfer between camera and PC is managed by an IEEE1394 interface card, supporting transfer rate of 800Mbps (IEEE1394b).

Software requirements

The camera is connected to a PC running Windows XP, which requires an installation of the manufacturer's device driver and SDK[4]. There are five different APIs included in the package, of which FireGrab has been selected. It has proven to be simple to use while still providing all required features.

This API controls important camera and card settings like transfer speed, auto focus or gamma correction. It allows for easy frame grabbing and quick conversion into data formats suited for display on screen.

The GUI has been implemented using FLTK[5]; a lightweight framework designed for simple but fast interfaces with a small memory footprint. It was chosen as it allows easy design and includes all required features without having a big impact on the overall program performance.

[3] Standard for high-speed serial connections
[4] AVT Fire Package version 2.8
[5] Fast Light Tool Kit version 1.1.x-r5940

Graphical User Interface

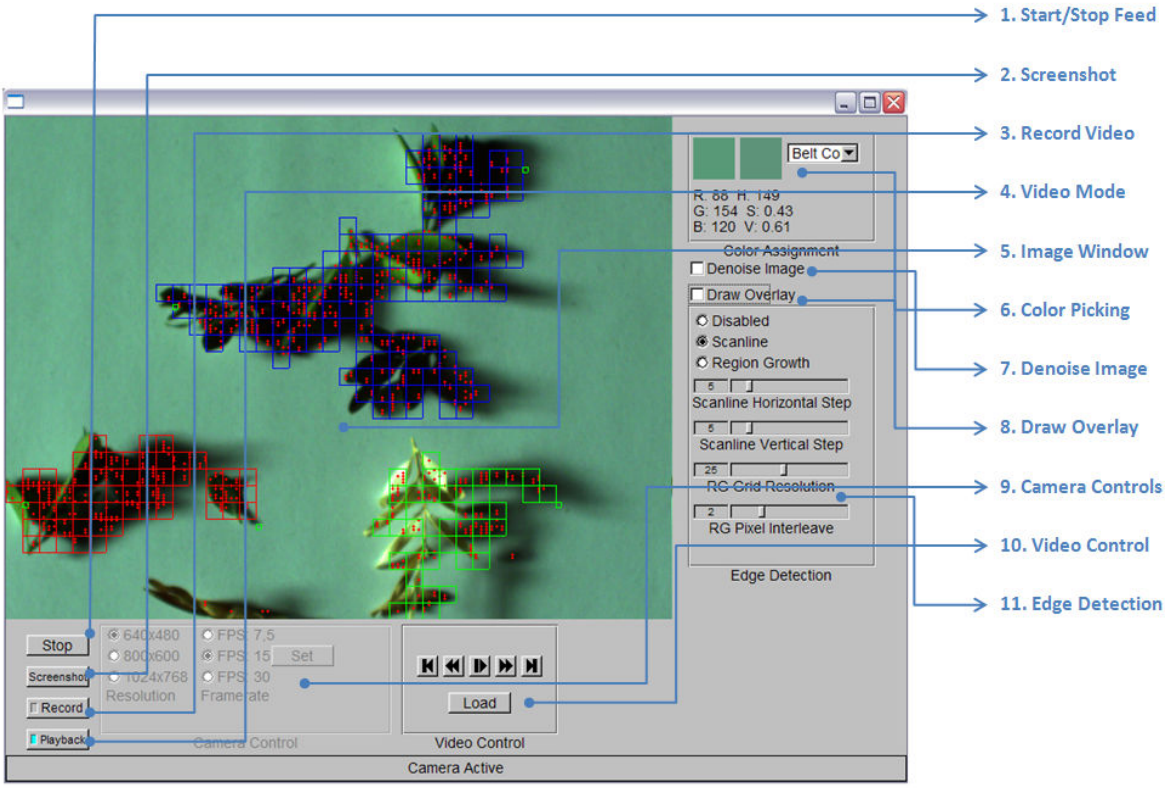


Image 5 - Screenshot of the GUI with feature description

The GUI has to provide numerous functions as a test platform for the object detection algorithms. Parameters for camera and plant recognition have to be adjustable and reproducible log-files of a camera feed need to be created.

The interface provides an image window (5), which displays a frame buffer that is either fed by the live camera feed or from a recorded video. Those are stored (3) as raw RGB[6] data and saved to a file. There is a switch to toggle between the two display modes (4). While in live feed mode, camera resolution and frame rate can be adjusted (9) and the feed stopped (1). In video mode we can select certain frames and play or pause the video stream (10). Furthermore, the current frame buffer can be saved as a PPM[7] file for debugging purposes (2).

[6] Additive color model with the channels red, green and blue
[7] Portable Pixmap – an uncompressed image format by Sun Microsystems

As described in the following chapter, the edge detection is working in the HSV color space[8]. Using the color-picking menu (6), colors in the image window can be selected and assigned to the different object classes. RGB and HSV values are displayed on the fly and the object hue values saved in a color table file. To verify the association the object color of each pixel can be displayed (8).

Various parameters of the edge detection are adjustable (11), which will be covered in more detail in chapter five. A simple noise reduction algorithm can be applied to the frame buffer before the edge detection is started (7). As this requires too much processing time it isn't recommended.

Several resulting parameters like iteration runtime, number of detected edge points or number of found objects are optionally printed to a console window which is launched together with the GUI.

[8] Color model focusing on accurately describing perceptual relations. Channels hue, saturation and value

5. Edge Detection

Overview

Edge Detection regarding machine vision is the process of finding the outlines of a scene object for later evaluation.

Edges in an image can be defined as regions wherein pixel values change significantly from their neighbors. This difference can happen in either image channel, e.g. different colors or changed intensity.

The methods used for finding these edges vary dependant on the input material. If the source image has only one channel a common approach is processing the image with convolution kernels. Those are a type of filter that calculates the value of a pixel by processing over its neighbors. Common convolution kernels for edge detection are Sobel, Prewitt or Canny. Each of those approximates discrete first order gradients with certain other operations included.

These methods can eventually fail as soon as the source has multiple channels, like the common RGB format, which has three color-channels. It is possible to determine the edges for each channel and combine them, yet there is no guarantee that edges exist for every color at the same position. Applying operations to all three channels eventually increases the processing time.

Considering the RGB example it generally isn't easy to identify similar colors by similar values. Objects are usually made of materials with discrete colors that can vary heavily in perception due to inconsistent lighting. In many cases a partially shadowed item has very different RGB triplets.

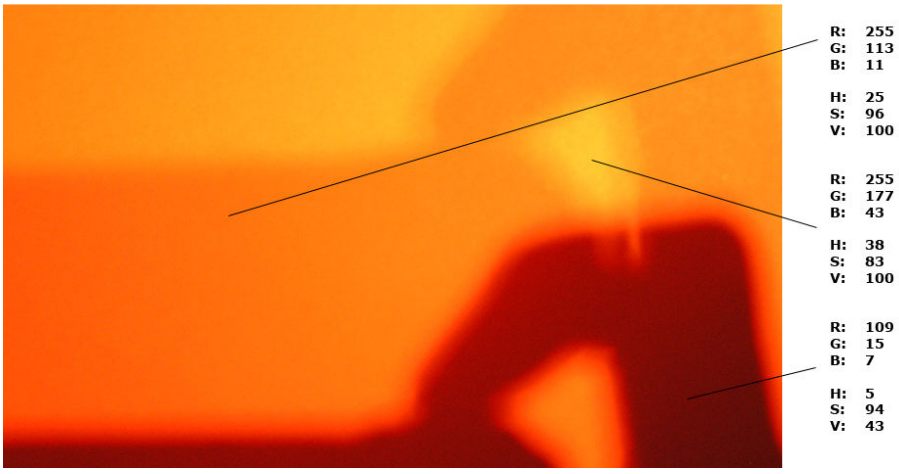


Image 6 - Uniform object partially shadowed with RGB and HSV values of certain points

Consequently it isn't easy to identify a material in RGB, if natural lighting is used. However there are color systems that fit the human perception in a more appropriate way while maintaining the same color diversity[9]. For this project HSV was chosen. Here a color is made up of the channels hue, saturation and intensity. Saturation and value define the strength and brightness of a displayed color. Hue is the angle of a color mapped on a circle as shown in the following picture.

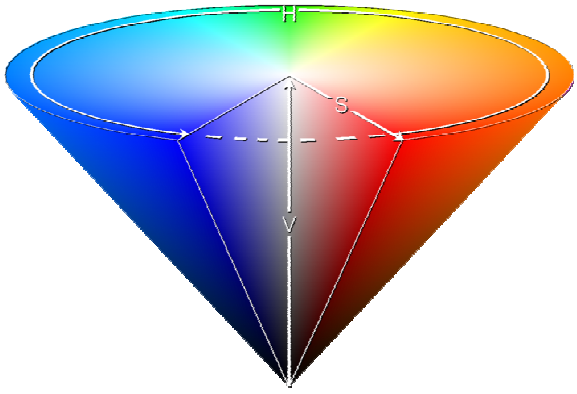


Image 7 - HSV color space cone with value as height, saturation as radius and hue as angle

As can be seen in image seven, colors with a similar angle are equally similar in human perception. Accordingly it is possible to identify materials by examining the hue channel only. Example values for an object can be seen in Image 11. While hue varies between 5 and 40, there are increased differences in the RGB values. As digital image processing is often based on this color system, it is necessary to perform an initial color space conversion before further processing.

The GUI offers the option to select pixels in a still image and assign hue angles to any object in the scene. These matches are stored in a simple integer lookup table using a precision of one degree, resulting in 360 possible values. The color space conversion from RGB to HSV is implemented with the algorithm of Gonzalez and Woods(Gonzales, et al., 2001).

However applying this algorithm to every pixel of the source image is a demanding task. To achieve real-time processing of images, it is necessary to reduce the amount of conversion operations. In this project two different methods have been implemented and are discussed in the following part of this thesis.

[9] Grassmann's first law of chromatics

Scanline algorithm

A simple procedure for edge detection is the scanline algorithm. It describes the simplification of a source image by using RLE[10] compression on certain lines. Multiple lines are divided into segments of a specific color. These segments are defined by start coordinate, end coordinate and color component. Consequently it can be used to identify the boundaries of an object in the scene, if it has a distinct color difference compared to its environment.

For this project the implementation allows to scan either horizontal or vertical with selectable line intervals, e.g. checking a row every fifth pixel. Another modification is the approximation of a linear image blur by analyzing segments of three pixels at a time and proceeding if two or more are of the same color. By default the application runs two iterations of the algorithm, analyzing in both mentioned directions.

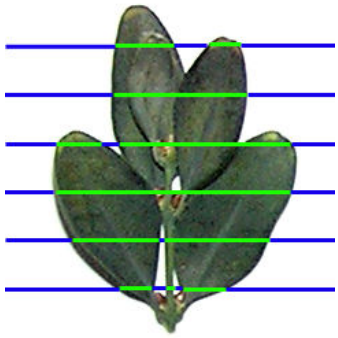


Image 8 - Example segments for scanline algorithm

[10] Run Length Encoding

Pattern-based region growth algorithm

The basic idea of this method is to examine certain image points for the color of the object to detect. If a match is found, the algorithm will analyze neighbor points until the object border is reached. Thus if the initial points are precise enough to find a start point in every object and every object pixel has correctly assigned hue value the algorithm is able to find a reliable edge.

In this application the first step is achieved by analyzing a rectangular grid of points of varying pixel interleave. From each point found the algorithm starts to advance in horizontal direction until first borders are found. Afterwards the algorithm checks several points in vertical direction based on the parameters of the previously found edge points. There are five potential candidates that can be checked for each pair of found points. A data structure was implemented which stores the current search direction and the previously detected edges. The following image shows how candidates are handled in detail.

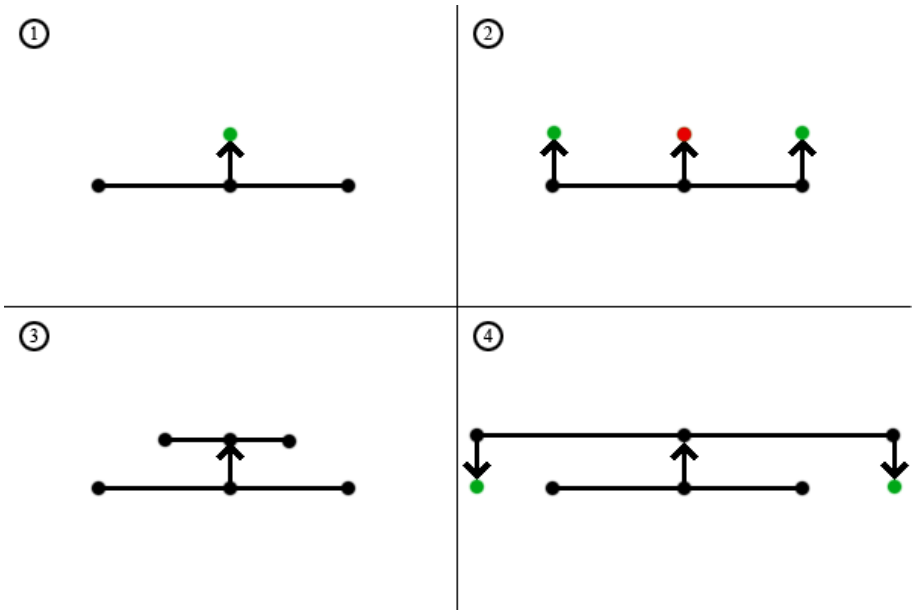


Image 9 - Region growth candidate handling

The first step is to determine the mean point of the current edge point pair and advance a step in the previously used direction to determine a new candidate point (1). If the result of the color table look up for that candidate point is negative another two candidate points are added by doing a horizontal step from previous edges (2). For any candidate point the algorithm grows horizontal until the next boundaries are found. In case (3) the candidate lookup for the first point has been

positive and the new line lies within the limits of the previous. In case (4) the new edge points exceed those edges and candidate points are added in the opposite direction wherever the line goes beyond. Following this strategy the region growth method is able to proceed in the majority of possible scenarios.

Furthermore, to prevent the algorithm from iterating endlessly a Boolean lookup array in image size was implemented. It stores successfully detected edge points. In case the color space conversion has a positive result and the matched pixel already has been detected, the iteration breaks.

The algorithm converges either as soon as all candidate points yield negative results or all edges have been detected. The point distance of the rectangular grid and the line interleave are adjustable in the program.

6. Recognizing Plants

Problems in Plant Recognition

The identification of plants in biological systematic usually is a difficult task. While some may share similar colors or shapes over the same genus and species they remain naturally grown objects. Therefore the recognition result is dependent on various factors influencing the plant in the growth process.

As described earlier, the target is to determine both position and orientation of the plant cutting in order to have the robot arm pick it up and place it into test tubes. The easiest way to achieve this is by identifying the lower and upper ending of the stem. Assuming the stem as a straight line proves to be a reasonable approximation of the real plants. Grabbing the middle of this assumed straight line, the robot should be able to pick up the plant with the required accuracy.

One of the main problems is the irregular result of the cutting process. In this project scions can have between 5 and 12 leaves and asymmetrical shapes. A few examples are demonstrated in the pictures below.

Image 10 – Different box tree cuttings

Another difficulty is the irregular positioning of the cuttings. The randomly placed plants aren't necessarily facing the fixed camera with their front. Consequently even the same piece can look very different if placed randomly as shown below.

Image 11 - Different stable positions of a single box tree cutting

Geometric Simplification

Finding a common model for all plants for determining the stem position has proven to be quite difficult. Three different models have been evaluated.

The first model is based on finding the longest line in a cutting. We assume the stem will be located along this straight or close to it. As most scions have a shape similar to the example plant below this generalization is justified in many instances. At the same time the center of gravity for the cutting is calculated. The border point of the line closest to that center is defined as the lower point of the stem. With the bottom leafs usually being bigger, this assumption is viable.

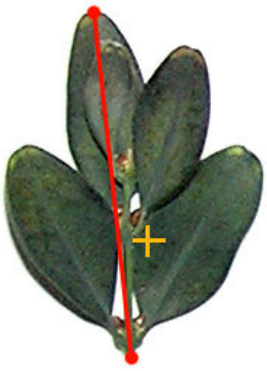


Image 12 - Longest line in a cutting with orange cross as center of gravity

The second model is based on detecting the quadrangle of maximum area in the cutting. Afterwards the median line through the smallest side and its opposite is calculated. This straight is extended to the boundaries of the plant. The intersection points with the edge are defined as stem end points, with the one being closer to the smallest side stored as upper end.

This model tries to simplify the overall cutting geometry by replacing it with a quadrangle. If the plant shape is simplified by ignoring form of the single leaves, it can be roughly approximated as the detected trapezium.

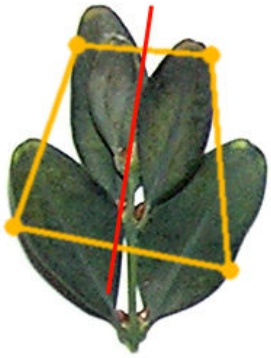


Image 13 - Maximum area quadrangle with median line as stem

The last model is based on points where two leafs join the stem. If it is possible to find two of those intersections in a plant a line through both can be calculated which resembles the original stalk very closely. The line ends at the cutting boundaries. Orientation is determined similar to the first model.

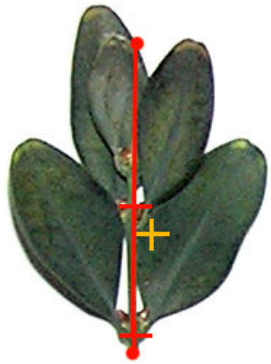


Image 14 - Intersections as red crosses with line extended to boundaries and center of gravity in orange

Any detected crossing corresponds directly to a point on the stem. If we are able to find two or more, a line through those intersections will be a very reliable approximation of the stalk.

Having tried all models the first is the most practicable. While the second model delivers results that are equally accurate it is far more computation intensive up to a point it exceeds the time available for real-time processing. The third model fails to deliver any good results, as the intersections are only visible in few plants.

Detection Quality

While the above models are able to detect a stalk fairly precisely, it is necessary to examine found lines for plausibility. Simple criteria are generated by idealizing the cutting as a triangle. Its height is assumed as stem. The center of gravity should lie on this line and separate it with a certain ratio.

The first criterion is the distance between the balance point and the stalk. A high distance indicates a highly asymmetric shape and is penalized.

The second criterion is the ratio between the distances of the stem end points to the center of gravity. In any isosceles triangle the factor is one to two, with the smaller side starting at the lower stem point. Any difference from this desired ratio affects the quality negatively, as it is assumed that proportions varying from the norm indicate an unevenly shaped plant.

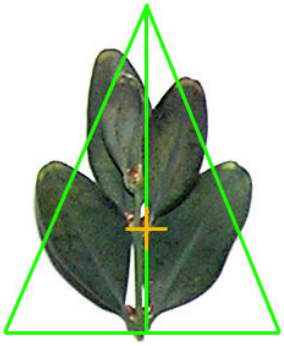


Image 15 - Idealized cutting with center of gravity and height as stem

7. Comparison

Theoretical runtime differences

As previously mentioned, the color space conversion is a very demanding operation. Even with region growth being an iterative operation, it is viable for big images to compare complexity on conversion operations alone.

The scanline algorithm complexity relies solely on the size of the source image. The number of conversion operations is calculated with the following formula.

$$O = n * \frac{w * h}{x}$$

Where *O* is the number of operations, *n* the number of algorithm cycles, *w* the image width, *h* the image height and *x* the line intervals mentioned above. For the example 640x480 input image, with horizontal and vertical runs, and a line interleave of 5 there have to be 122.880 conversions performed.

Determining the region growth complexity is more difficult, as it relies on the result of the initial grid checkup. It scales linear with the amount of pixels belonging to objects in the scene. The number of conversion operations is calculated as follows.

$$O = \frac{w * h}{g^2} + \frac{\sum_{i=1}^{k} p_i}{x}$$

The first part is the initial grid step with *g* being the grid resolution. The second is the number of color conversions in the iteration algorithm. The amount of objects is *k* and p_i the number of pixels in a single object. If our 640x480 test image contains 10 objects that have an average of 5.000 pixels each, 37.288 color conversions have to be performed. The grid interleave was 5 pixels and the line interval 2 pixels.

Finally in the current setup the region growth algorithm should perform better than the scanline algorithm, as the scene is sparsely filled with fairly big objects that are hardly ever missed by the initial grid.

Theoretical quality differences

Criteria for edge detection quality are the amount of edge points found, and if those candidates are part of the object edge or can be attributed to noise or undesired particles of the same color.

The full quantity of edge points is found when every image pixel is getting analyzed. If it is assumed that the objects are uniformly distributed in the image, following approximations are viable. For the scanline algorithm:

$$P \approx \frac{N}{x}$$

In this formula *P* is the number of found edge points, *N* their overall amount and *x* the line interleave setting. The region growth method requires a deduction of undetectable objects, which are not registered by the initial grid. The additional parameter *f* is the percentage of initially found objects in relation to the overall number of objects.

$$P \approx f * \frac{N}{x}$$

Consequently scanline theoretically is able to find more edge points, unless the starting grid for region growth is sufficiently precise, leading to a value $f = 1.0$.

Reducing edge points generated by noise can either happen before the detection itself, using a generic noise reduction algorithm, or during the recognition process. While the first method generally leads to a superior result, it consumes an unreasonable amount of computation time. The noise reduction integrated in the scanline algorithm is an imprecise approximation of a linear blur. Evaluation has shown that the impact on performance is minor while the result remains satisfactory.

Undesired particles of the same color are hard to eliminate during the edge detection algorithm. Their effect on both methods is similar. Run time is increased and object recognition might fail. As the BioRob project is working in a closed environment it is assumed that no alien materials will occur. Consequently they are ignored.

Concluding it can be said that a better quality is to be expected from the scanline algorithm in its current form. With reasonable line interleave settings the risk of missing objects is smaller compared to the region growth algorithm. Furthermore the implemented noise reduction yields good results without effecting processing time severly.

Measured differences

Every test has been performed with a 640x480 resolution test video which was recorded at 30 frames per second. The scene consists of five detectable objects. A suited color table has been used throughout all runs. Any final result is the mean of three independent measurements. Full result tables are available in the appendix.

The test system was a PC equipped with a 1.5 GHz dual-core AMD CPU with two gigabytes of RAM running Windows XP SP2. Background processes have been cancelled as far as possible.

The first chart displays the duration in milliseconds, the number of detected points and the number of found objects for the scanline algorithm with different line interval settings. It is noticeable, that the number of detected point grows linear to the required time. The maximum run time, the object detection might require, is limited by desired real-time processing. At 30 fps all operations have to be finished within 33 milliseconds. At line interval setting two the scanline algorithm almost exceeds those limits. It is likely that the available will be overstepped on more complex scenes. At high line interval settings it can occur that the number of found objects is higher than visible plants. This can be attributed to the edge points being distributed sparsely which eventually lead to a single plant being split in two.

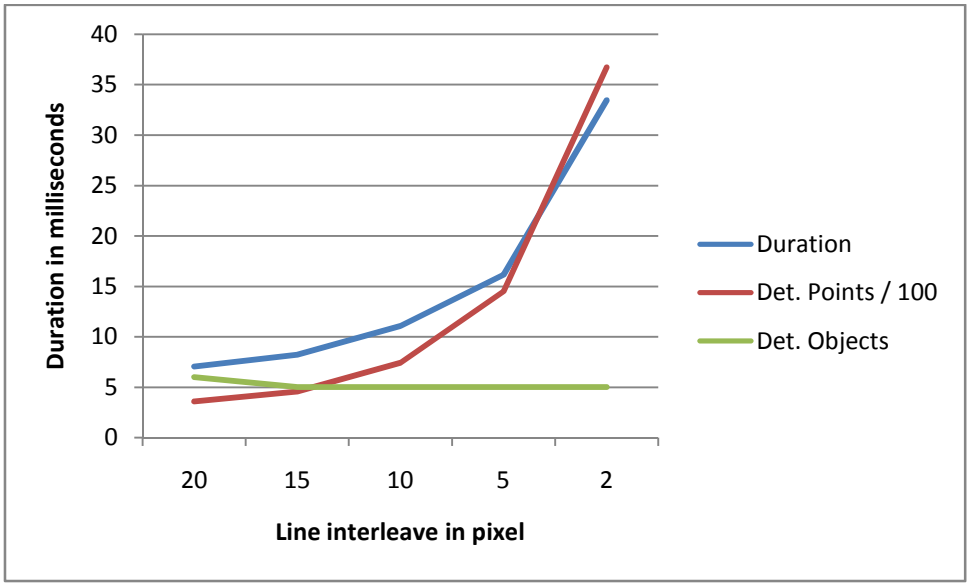


Image 16 - Duration and detected points for the scanline algorithm at various line interleave settings

The second chart shows the same parameters for the region growth algorithm. Obviously the number of detected points is higher in general while the maximum duration never approaches the available time. The object splitting that eventually appeared using the scanline algorithm does not occur due to line interleave settings generally being lower.

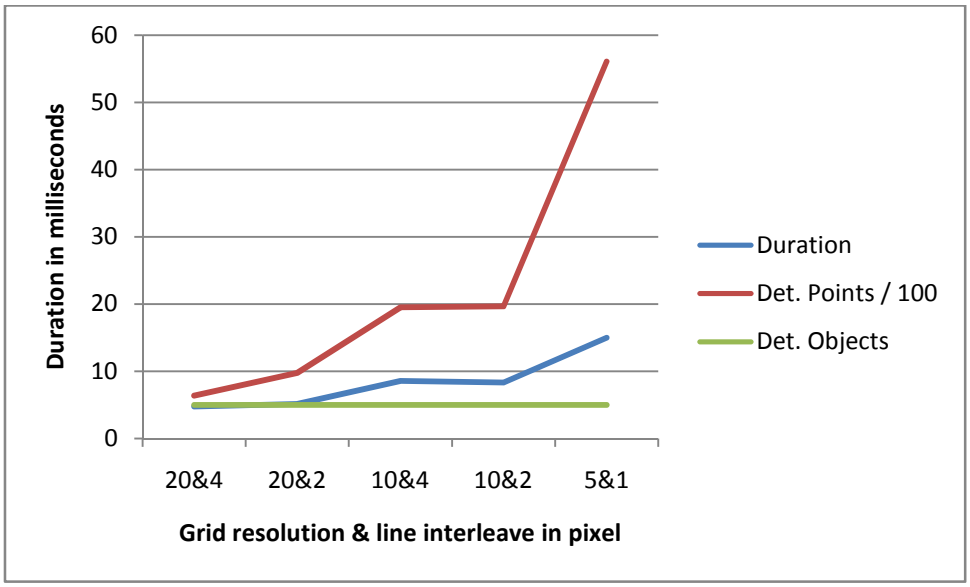


Image 17 - Duration and detected points for the region growth algorithm at various (grid resolution & line interleave) settings

Result and consequences

Conclusively the results meet the expectations for the different approaches. While scanline yields good detection and a subjectively better shape quality, its performance suffers due to the large amount of conversion operations. Being viable in low-resolution projects, only optimized versions running on powerful modern PC systems are sufficiently fast for real-time applications at high frame rates.

The designated resolution for the BioRob project is 1024x768, running at 30 fps. The amount of pixels that are processed is approximately 2.5 times larger than the amount that has been analyzed in these tests. Extrapolating the duration of the algorithms to the desired resolution usnig line interleave 5 for scanline and 10 & 2 settings for region growth, a processing time of 40 ms for the first and 21 ms for the latter can be expected

Evaluating these results only the region growth algorithm offers edge detection efficient enough to be run on common PCs without much optimization. While the shapes are looking less sophisticated, the resulting objects are equal to those generated by the scanline method.

8. Conclusion

While machine vision and industrial robots are common in international and larger national companies, the pricing for enterprises working with small quantities remains too high. In this project it was shown that using a reasonable priced camera system and a cheap desktop PC system tasks can be achieved that have been restricted to high-priced, specialized systems a decade ago.

However the results aren't completely satisfying. Particularly the stem detection algorithms can be improved in the future. More complex recognition methods will become usable on basic systems, e.g. adaptive shape databases, which compare the boundary generated from recognized edge points to an extendable library of shapes. As of now the required processing time exceeds the available limit for real-time applications.

Tasks that could not be completed in the available time for this thesis are the integration and adaption of the object recognition into the control software of the robot. Real world testing and optimization has yet to be performed. The current modular implementation will allow those tasks and integration of new functionality in a short period of time.

The recognition process in its current form is based on generic algorithms and hence lacks optimization. With the advent of multi-core CPUs and general-purpose GPUs drastic improvements can be achieved for certain parts of the detection, without increasing the overall cost of the system. Operations, such as color space conversion can be highly parallelized. This might allow future implementations to process higher camera resolutions or increased frame rates, while remaining real-time.

The BioRob project and similar research can help to extend the robot and machine vision industries into new markets. Small, flexible and cheap systems have the potential to revolutionize small and medium scale manufacturing in a similar way high-quantity and high-tech industries progressed the last three decades.

9. Resources

Bibliography

Aktive Wärmefluss-Thermographie zur Qualitätssicherung [Online]. - March 21, 2008. - http://www.vision.fraunhofer.de/de/2/projekte/231.html.

Allied Vision Technologies Marlin Series [Online]. - March 21, 2008. - http://www.alliedvisiontec.com/avt-products/cameras/marlin/f-080-b-bs-c.html.

AXTEL Machine Vision Resources [Online]. - March 21, 2008. - http://www.axtel.com/machine-vision.html.

Cognex - Company History [Online]. - March 21, 2008. - http://www.cognex.com/CognexInfo/FastFacts/default.aspx?id=286.

Digital Image Processing [Book] / auth. Gonzales R.C. and Wintz P.A.. - [s.l.] : Longman Higher Education, 2001. - ISBN 978-0201110265.

Intelligent Vision Systems for Industry [Book] / auth. Batchelor B.G. and Whelan P.F.. - [s.l.] : Springer-Verlag, 1997. - ISBN 3-540-19969-1.

InVision 5000 Color Sorting System [Online]. - March 21, 2008. - http://www.compacsort.com/webfiles/CompacSort/files/InV5K_Eng.pdf.

Machine Vision Online Archived Articles [Online]. - March 21, 2008. - http://www.visiononline.org/public/articles/articles.cfm?cat=19.

Nikon Online Resource for Microscopy Education [Online]. - March 21, 2008. - http://www.microscopyu.com.

Olympus Wafer Inspection System MX61A [Online]. - March 21, 2008. - http://www.olympusmicroimaging.com/index.cfm/page/products.index.cfm/cid/1593/navid/265/parentid/5.

Pattern Recognition Archive [Online]. - March 21, 2008. - http://cgm.cs.mcgill.ca/~godfried/teaching/pr-web.html.

Systematik und Taxonomie: Methoden und Regeln zur Klassifikation von Pflanzen [Online]. - March 21, 2008. - http://www.biologie.uni-hamburg.de/b-online/e43/43.htm.

10. Appendix

Measurements tables

Table 1 - Scanline edge detection with line interleave 20

	Measurement 1	Measurement 2	Measurement 3	Mean
Duration	6,6813	7,2177	7,2314	7,0435
Det. Points	356	356	356	356
Det. Objects	6	6	6	6

Table 2 - Scanline edge detection with line interleave 15

	Measurement 1	Measurement 2	Measurement 3	Mean
Duration	8,1471	8,2770	8,2321	8,2187
Det. Points	454	454	454	454
Det. Objects	5	5	5	5

Table 3 - Scanline edge detection with line interleave 10

	Measurement 1	Measurement 2	Measurement 3	Mean
Duration	11,1198	11,0279	11,0601	11,0694
Det. Points	740	740	740	740
Det. Objects	5	5	5	5

Table 4 - Scanline edge detection with line interleave 5

	Measurement 1	Measurement 2	Measurement 3	Mean
Duration	16,2956	16,1613	15,9744	16,1437
Det. Points	1449	1449	1449	1449
Det. Objects	5	5	5	5

Table 5 - Scanline edge detection with line interleave 2

	Measurement 1	Measurement 2	Measurement 3	Mean
Duration	33,4445	33,7582	33,1028	33,4352
Det. Points	3673	3673	3673	3673
Det. Objects	5	5	5	5

Table 6 - Region growth edge detection with grid resolution 20 and line interleave 4

	Measurement 1	Measurement 2	Measurement 3	Mean
Duration	5,0411	4,7729	4,4628	4,7589
Det. Points	636	636	636	636
Det. Objects	5	5	5	5

Table 7 - Region growth edge detection with grid resolution 20 and line interleave 2

	Measurement 1	Measurement 2	Measurement 3	Mean
Duration	5,2152	5,1688	4,9886	5,1242
Det. Points	972	972	972	972
Det. Objects	5	5	5	5

Table 8 - Region growth edge detection with grid resolution 10 and line interleave 4

	Measurement 1	Measurement 2	Measurement 3	Mean
Duration	8,7380	8,6374	8,3371	8,5708
Det. Points	1948	1948	1948	1948
Det. Objects	5	5	5	5

Table 9 - Region growth edge detection with grid resolution 10 and line interleave 2

	Measurement 1	Measurement 2	Measurement 3	Mean
Duration	8,2756	8,2251	8,3782	8,2930
Det. Points	1966	1966	1966	1966
Det. Objects	5	5	5	5

Table 10 - Region growth edge detection with grid resolution 5 and line interleave 1

	Measurement 1	Measurement 2	Measurement 3	Mean
Duration	14,6493	14,6935	15,5154	14,9527
Det. Points	5606	5606	5606	5606
Det. Objects	5	5	5	5