Real-time Detection of Young Spruce Using Color and Texture Features on an Autonomous Forest Machine

Heikki Hyyti, Jouko Kalmari, and Arto Visala

Abstract—**Forest machines are manually operated machines that are efficient when operated by a professional. Point cleaning is a silvicultural task in which weeds are removed around a young spruce tree. To automate point cleaning, machine vision methods are used for identifying spruce trees. A texture analysis method based on the Radon and wavelet transforms is implemented for the task. Real-time GPU implementation of algorithms is programmed using CUDA framework. Compared to a single thread CPU implementation, our GPU implementation is between 18 to 80 times faster depending on the size of image blocks used. Color information is used in addition of texture and a location estimate of the tree is extracted from the detection result. The developed spruce detection system is used as a part of an autonomous point cleaning machine. To control the system, an integrated user interface is presented. It allows the operator to control, monitor and train the system online.**

I. Introduction

Field and service robotics have made a considerable impact in agriculture and forestry industries. Many advanced robotic solutions have been proposed in the area. For example, a prototype of a legged forest machine was already developed in the late 90's by John Deere [1]. Recently, the motivation to mechanize silvicultural operations has increased in the Nordic countries and several new forest machine concepts have been developed [2]. Our current research focus is to develop a new concept of autonomous cleaning of young spruce trees.

Silvicultural cleaning is used to relieve chosen conifers from competing, naturally regenerated, broadleaved trees [3]. The current solution to clean the surroundings of young spruce is either manual work or using mechanized human operated machines. In addition, the cleaning operation can be carried out chemically. However, it has some environmental concerns and a strong public opposition [4]. Point cleaning is a practical precommercial thinning method, in which the cleaning is performed within a certain radius around the main stem [5].

Manuscript received April 30, 2013. This work was done under the NeoSilvix project, which was funded by TEKES, Finnish Funding Agency for Technology and Innovation and participating companies.

H. Hyyti is a PhD Student with the Autonomous Systems research group, Department of Automation and Systems Technology, School of Electrical Engineering, Aalto University, PO Box 15500, FIN-00076 Aalto, Finland (e-mail: heikki.hyyti@aalto.fi).

J. Kalmari is a PhD Student with the Autonomous Systems research group, Department of Automation and Systems Technology, School of Electrical Engineering, Aalto University, PO Box 15500, FIN-00076 Aalto, Finland (e-mail: jouko.kalmari@aalto.fi).

A. Visala is a Professor with the Department of Automation and Systems Technology, School of Electrical Engineering, Aalto University, PO Box 15500, FIN-00076 Aalto, Finland, (e-mail: arto.visala@aalto.fi).

In order to operate in natural forest environment, the autonomous forest machine requires advanced sensors and methods to detect the target spruce tree among other vegetation. Texture and color features could be used to separate spruce from other vegetation. There is a lot of variation in the environment and a single texture or color method may not be sufficient for the task. Multiple methods can be used together to acquire enough different features from the image to detect the target reliably. The drawback is that a competent detection algorithm can be computationally highly complex to solve. In order to use texture analysis and detection methods in a closed loop controlled system, the algorithm should be computed relatively fast and in deterministic time.

In order to detect spruces among other vegetation, texture detection methods by Jafari-Khouzani et al. were tested [6, 7]. They claim that their rotation invariant multiresolution texture analysis using Radon and wavelet transforms is more robust to additive noise than Log-polar wavelet signatures or multichannel Gabor filtering. Our initial results for spruce detection were successful. Although the algorithms are computationally complex, they were selected for real time implementation. According to [8], the Radon transformation for an image of size 128x128 on a work station computer with 3.06GHz and 2GB RAM takes 482ms, and their contending methods take 543ms and 159ms for the same image. This computation time for a single image block is too slow for a real-time system requiring hundreds of image blocks to be analyzed for a single frame.

The development in graphics processing units has made many computationally expensive machine vision algorithms available in real-time applications. This paper presents an efficient CUDA GPU implementation for Rotation-invariant multiresolution texture analysis using Radon and wavelet transforms by Kourosh Jafari-Khouzani and Hamid Soltanian-Zadeh [6]. The texture detection method is improved using additional color features. In addition, a GPU implementation of k-NN feature detector is used to classify image blocks. An estimated location of a tree is extracted from the classification results. The location is used to control a crane of a forest machine.

Autonomous point cleaning is performed by our research platform that uses a prototype of a point cleaning tool. The tool is attached to a forest crane which is attached to an agricultural tractor. Together these components form a small-scale forest machine which is depicted in Fig. 1.

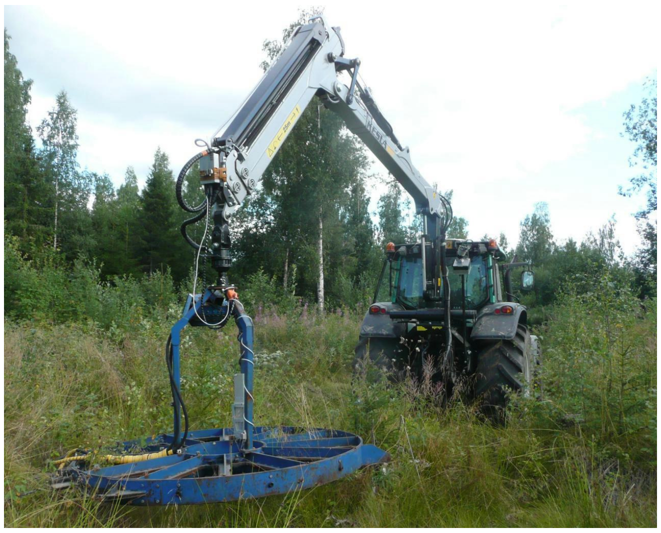
Fig. 1. The small-scale forest machine with a point cleaning tool in realistic environment. Young spruce trees are searched among other vegetation using a camera attached to the point cleaning tool and a machine vision system dedicated to the task.

II. Hardware and Environment

The small-scale forest machine used in the experiments consists of an agricultural tractor, Valtra T132, and a forest crane, Kesla 305T. The prototype point cleaning tool was made by Pentin Paja. The agricultural tractor and the forest crane are designed to work as a research platform for an autonomous forest machine. The forest crane is instrumented for autonomous operation [9]. The prototype point cleaning tool is depicted in Fig. 2. It is used to remove competing vegetation around the target plant by lowering the tool on the target and cutting other vegetation around it using hydraulic-operated blades.

Aiming the point cleaning tool on the target tree is difficult as the tool is freely hanging on the tip of the crane far from the cabin. Therefore, a camera lift has been developed to enable a camera to be attached to the point cleaning tool. Using the camera lift, a wide scene can be imaged through the central hole. When the tool is lowered on the target to cut the surrounding vegetation, the camera is lifted up enclosing the camera to a sealed metal box protecting it from hitting the tree. The camera lift is shown closed in Fig. 1 and open in Fig. 2. A typical view of the camera is shown in Fig. 3. In images Fig. 2 and Fig. 3, the cleaning tool is approximately in the same place.

The camera attached to the cleaning tool is DFK 41AU02 color camera made by The Imaging Source. Camera data is collected in raw format at 15 fps using a resolution of 1280x960 pixels. A lens with a focal length of 6.5mm is used to acquire a wide field of view. The camera shutter and gain are regulated using a custom automatic controller which is controlling the amount of overexposured pixels. The idea of control is simple; the amount of pixels with the highest possible intensity value is used as a reference when shutter time and gain are controlled. Simultaneously, the gain is minimized and the shutter time is limited to a predefined maximum value to avoid motion blur. Automatic gain and shutter control are crucial while working outdoors, as brightness is varying and robust computer vision is required. In addition, the object detection method has to cope with shadows and changing illumination.

Operational tests demonstrating the behavior of the system were run on a small test field with five planted spruces and a set of deciduous trees and high grass complicating the task. The test field is shown in Fig. 2 and Fig. 3. In comparison, Fig. 1 shows the system in a more realistic scene than our test field is.

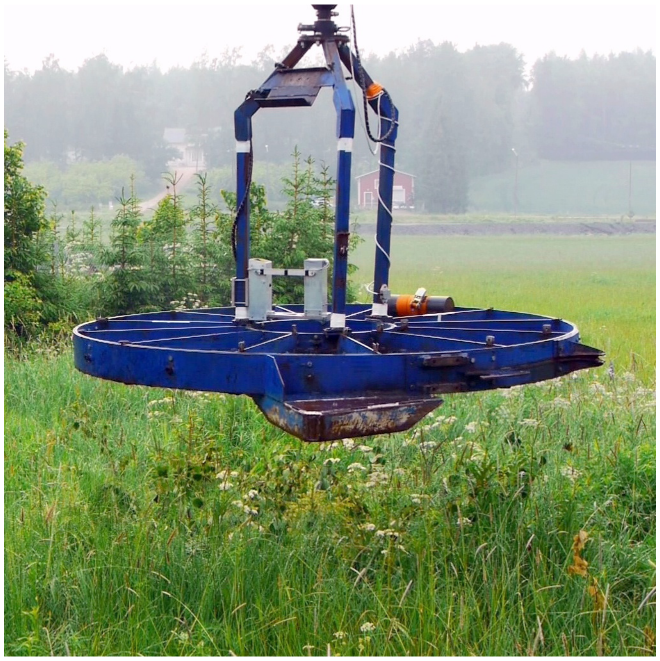
Fig. 2. The point cleaning tool is in operation on a test field. The camera lift is opened to allow camera to shoot through the central hole. The autonomous forest machine was demonstrated on a test field of a few planted spruces among other vegetation.

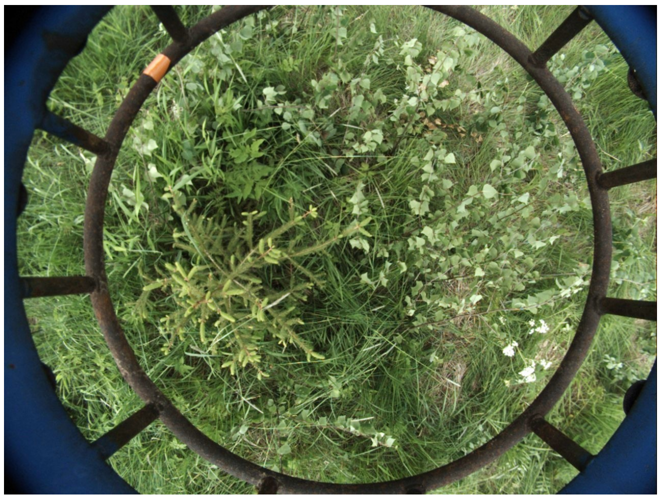
Fig. 3. Camera image used to search young spruce trees. The camera is shooting through the central hole of the point cleaning tool. The tool is approximately at the same place than in Fig. 2.

III. METHODS

A. Rotation-invariant Radon and wavelet features

Texture analysis is done only on the intensity image. To reduce the computational load, the image is first cropped so that the visible parts of the point cleaning tool are left out. The image is then divided into smaller blocks that are processed and classified individually. Block sizes of 32x32, 64x64 and 128x128 pixels were used. The blocks are always overlapping each other by half of the block size, i.e. when using blocks with size 32x32 pixels; they overlap their neighboring blocks by 16 pixels each. Thus, the texture analysis is done for a grid with nodes 16 pixels apart.

The texture analysis is based on a method proposed by Jafar-Khouzani et al. [6] that uses Radon and Wavelet transforms for creating rotation invariant analysis. Rotation invariant features are chosen because the viewpoint is directly from above and young spruce trees are roughly radial symmetric. A circular hard limiting windowing function is used to shape the image block radial symmetric. A sample block of 64x64 pixels selected from the intensity channel of image in Fig. 3 is shown in Fig. 4A. The selected block is transformed using Radon and translation-invariant wavelet transforms.

The Radon transform is a two-dimensional transform. Each pixel in the transformed image is the sum of pixel intensities along a line at a given angle and a distance from the center of the image. The angle is mapped on the x-axis and the distance on the y-axis in the transform result which is shown in Fig. 4B. The Radon transform of an image block is normalized using a transform of the windowing function itself. Rotation in the original block is seen as translation along the x-axis in the Radon transform.

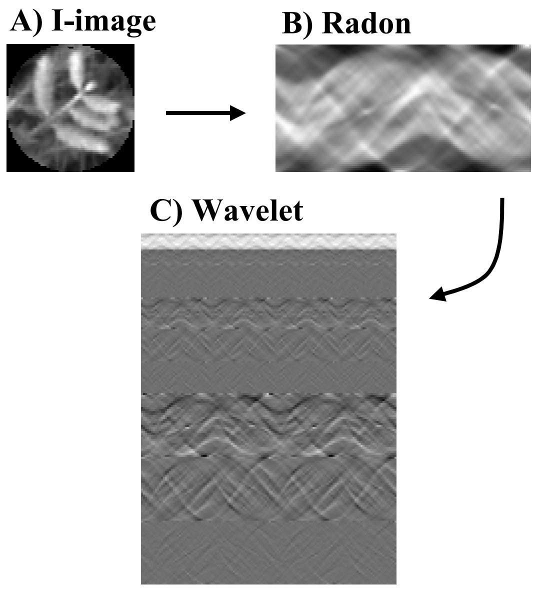

Fig. 4. The texture feature computation procedure in three steps: A) an intensity image block of size 64x64 pixels with a windowing function, B) a Radon transform computed from A, and C) a wavelet transform computed from B.

According to Martin Brady [10], for image of size NxN, majority of Radon transforms have computational complexity of $O(N^3)$. His discretized version has complexity of $O(N^2 \log N)$. However, we implemented the standard, non-discretized version of the Radon transform for simplicity.

The wavelet transform is used to decompose the Radon transformed image to different frequency bands. As the goal is to use rotation-invariant texture analysis, translation in the x-direction in the Radon image should not affect final features. Jafar-Khouzani et al. [6] proposed that instead of using a normal 2D-wavelet transform, a translation invariant wavelet transform [11] should be used in the x-axis and normal wavelet transform in the y-axis. The Haar wavelet is used for its simplicity. We used three wavelet levels causing nine sub-bands and a residual. They are all depicted in Fig. 4C. High frequency bands are on the bottom and the low frequency on the top.

B. Color features

It is intuitive to use color information in aid of texture in spruce detection algorithm. In the article by Yu-Ichi Ohta et al. [12] about color information for natural image region segmentation, they conclude that the best color features for segmentation are $I1 = (R+G+B)/3$, $I2' = (R-B)$, and $I3' = (2G-R-B)/2$. In their study, the three features were significant in this order. These same orthogonal color channels were later successfully used by Brian Steward and Lei Tian in real-time weed detection [13]. They called channels to I (intensity), RB (redness or blueness), and EG (excessive green) respectively, forming an EGRBI color transform. The color channels are presented in Fig. 5 along with the original RGB color image. The image in the figure is cropped from the example image in Fig. 3.

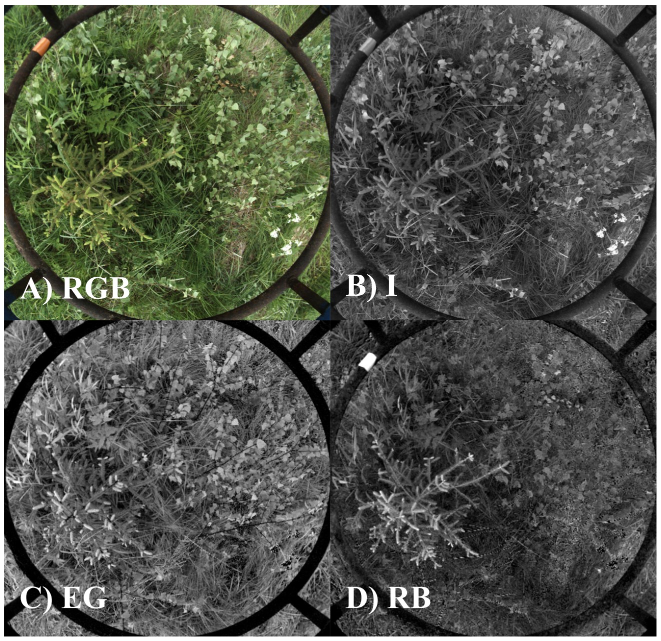

Fig. 5. The color features are calculated from EGRBI transformed image, which consists of Intensity (B), Extensive Green (C), and Red-Blue (D) channels. The original image is depicted in (A).

As depicted in Fig. 5, the spruce can be visually detected from the channels of EGRBI color transformed image. Especially the RB image usually shows spruces with brighter intensity than surrounding plants. In addition, spruce shoots are brightly visible in the EG channel.

As the field is imaged in natural light, there usually are shadows in the image. In addition to the natural shadows, the point cleaning tool casts shadows over the plants on a sunny day. In the natural images, EG and RB channels are usually correlated with the intensity channel. Therefore, EG and RB color channels were normalized pixel-vise by dividing EG and RB channels with the I channel. These new color features $EG' = EG / I$ and $RB' = RB / I$ were averaged around the same image blocks used in the texture analysis.

C. Feature extraction and k-NN classifier

The texture features used in the classification are calculated using the mean of square roots of absolute values for each sub-band in the wavelet transform. These features were used because Jafar-Khouzani et al. [6] found out that it gave better results compared to traditional energy and uniformity measures.

Training of the classifier is done with images selected using the user interface. The operator marks the spruce tree from the training image with two circles. The inner circle defines a zone where everything is certainly a part of spruce. All features outside the second circle are considered as background i.e. non-spruce. The training data is constructed as a set of features extracted and labeled using these images and circles labeling the training data.

Classification is done using the k-NN classifier with Euclidian distance to classify each image block into an appropriate class. In this algorithm, an unknown sample is assigned to the class most commonly represented in the collection of its neighborhood [14]. The k-NN classifier neighborhood size, k was chosen to have value 7. Instead of sharp classification result, the classifier is used to calculate the number of votes for classes. These votes are later used to find the estimate for the location of the spruce tree.

D. CUDA GPU computation

To realize the machine vision algorithms in real time, the vast parallel computation power of Graphic Processor Units (GPUs) was utilized. CUDA i.e. Compute Unified Device Architecture, an architecture developed by Nvidia was used. The GPU is used to calculate the Radon and wavelet transforms and to extract features. Also, the k-NN classification is implemented on CUDA.

CUDA optimization requires a lot of testing and tuning. There are three basic strategies to increase the performance of the CUDA code [15]: increasing the parallel execution, optimizing memory usage and optimizing instruction usage. The CUDA platform has profiling tools to examine the performance of different implementations.

Inside the GPU, there are a number of parallel processor cores; our laptop computer has 192 of them. Each of these cores can run many so called threads. These threads are not similar to a thread on a CPU. Each GPU thread is running the same kernel at the same instruction. As switching between these threads is very fast, it is often wise to break down the algorithm to thousands of small threads that are run parallel and consecutively. For example, other threads can continue to next instruction while one still waits for the data from memory.

Data transfer between CPU and GPU is often a bottleneck in total system performance. Therefore, it is wise to load the input data to the GPU memory, do multiple consecutive tasks on the GPU and load the end result back to the CPU. There are multiple types of memory inside the GPU: global memory, local memory, shared memory and texture memory. Each of these memory types has their advantages and limitations.

We decided to use texture memory to store the image data used as input to the Radon kernels. Accessing texture memory is quite fast and the interpolation required in the Radon transform can be achieved automatically. In the Radon transform, different threads inside a CUDA block are calculating different orientations for the same image block, and therefore the texture caches can be utilized effectively.

Optimizing instruction usage consists mainly of minimizing expensive instructions and avoiding branching in the code. Trigonometric functions are much more expensive than normal arithmetic operations. Branching causes some of the threads to be paused temporally on a single core until the code paths converge again. Therefore branching should be avoided and all parallel kernels should have similar code paths.

As implementing algorithms efficiently on the GPU requires more effort and experimentation than on a CPU, it is not reasonable to do everything on the graphics card. Therefore, we have implemented only the most expensive algorithms on the GPU. These algorithms were: the Radon transform, the wavelet transform, the feature extraction, and the K-nearest classification. Limited effort was expended on optimization, so further gains in efficiency are possible.

E. Reference CPU implementation

The Radon and wavelet transforms and the feature extraction were implemented in a single thread CPU-code as well. These algorithms were used as a reference for GPU computation. The CPU implementation was accomplished using OpenCV library. The Radon transform for CPU was also optimized by rearranging data for more optimal memory access. This allowed the use of fast arithmetic operations on arrays in OpenCV. This optimization works well with multiple similar parallel blocks, which are computed simultaneously. Matlab has a built in Radon transform function. Our CPU and GPU implementations were compared to the Matlab implementation to get a comparable reference for the computational power of our computer.

F. *Detection method in real-time control loop*

In order to use texture and color classification result to visual servoing the forest crane, the target location and detection quality of the target has to be extracted from the classification result. We estimated the target spruce location by first counting the number of spruce votes for every classified image block in the grid. The voting was done in the k-NN detector by counting spruce detections from the seven nearest neighbors in the features extracted and labeled from the training data. The maximum number of spruce votes is therefore seven and the minimum is zero.

The location of the target is estimated from the classification results by calculating a median for the votes in x and y directions separately for the whole grid. Only grid cells with votes more than a set threshold were counted in. A threshold of spruce votes larger than three was used to filter out non-spruce detections and to take into account only the votes with most spruces in k-NN neighborhood.

The median gave a robust estimate of a center location of a large blob of high spruce-voted image blocks. The actual location was refined by using a weighted average over spruce votes near the median location. The quality of the tree detection was estimated by counting the sum of spruce votes inside a circle centered to the estimated location. The circle size was tuned to correspond to an average target tree.

If there were enough total votes inside the given circle compared to the sum of all votes in the grid, the spruce tree was detected. Because of the comparison between the blob and the surroundings, the detection was prevented when there were too much spruce votes in the grid. Only relatively small-sized high-spruce-voted blobs were assumed to represent real spruce detection.

To filter out random detections around the image, a heuristic dependency between adjacent detections was introduced. To actually detect a spruce tree, we required the detection to occur in two adjacent frames spatially close together. This requirement prevented any random detection only from a single frame.

The final detection result was used in visual servoing the forest crane and the point cleaning tool. Although our algorithms are fast, there is still substantial delay between the image acquisition and the detection result. The delay was minimized by using custom data transfer protocols and software to acquire and transmit the camera image to the computer controlling the system. The image acquisition and transfer times were estimated with a high accuracy. The pose of the forest crane was recorded and the pose of the camera was estimated at the time of image acquisition. The crane was controlled using the estimated relative pose of the target spruce.

The automatic point cleaning was performed by first scanning through the field by using a predesigned path. When the spruce was detected, the tool was controlled over the center of the detected tree and lowered down after the swaying was damped. The actual cutting operation was not performed in our test as the hydraulic actuator was removed from the tool. After the operation, the tool was lifted and the search for a next spruce continued.

G. *Data visualization in user interface*

A user interface was developed for controlling and monitoring the forest crane and to train and test the spruce detection system. The user interface depicted in Fig. 6 shows the real-time camera image from the point cleaning tool and draws the detection result over the image. The point cleaning tool is attached to a rotator that allows it to rotate around its axis. The camera image is rotated on the screen to stabilize image in spite of the rotation of the tool. All parameters affecting to the image and spruce detection can be changed online.

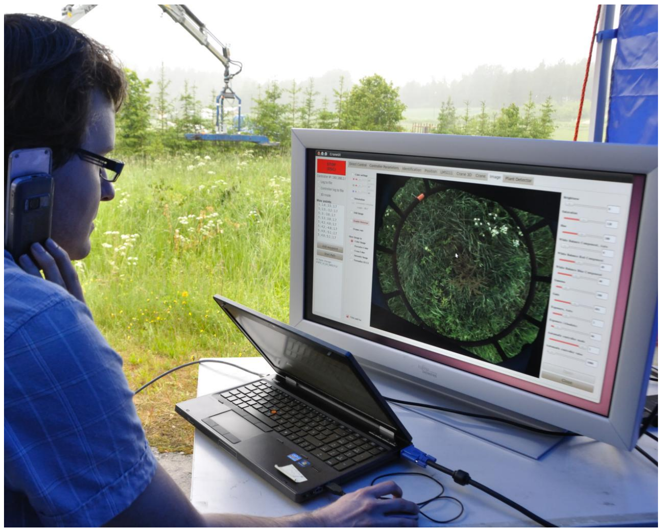

Fig. 6. A user interface for training the classifier and tracking young spruce trees. The same interface is used to remote control the forest crane. The controlled forest crane and the tool are shown in the background.

The detection result was drawn on the real-time image using colored dots. The size of a dot corresponds to the amount of spruce votes around the current image block. The final detection result (see section III.F) was shown to the driver with a blue circle around the target tree. In remote controlling mode, operator can drive the crane over a certain spot by clicking it on the screen. Similarly, the operator can label new training data online to the spruce detector (see section III.C). In the demonstration, the user interface was used outside the cabin as shown in Fig. 6.

In practice, the k-NN classifier can be trained online by driving the point cleaning tool over a young spruce tree and selecting it with a few mouse clicks. The left click is setting a center point of an inner circle to include all the features inside to a spruce class. The right click is setting a larger circle to include the surrounding points to a non-spruce class. The mouse wheel is used to scale both of the circles.

Using this method to collect training data from the environment, a few spruces can be easily trained for the classifier. By this way, the training data and the classified images are more similar than by collecting the training data set beforehand possibly in different conditions or illumination.

IV. Results

The detection algorithm was implemented and optimized for GPU and CPU separately. A single frame consisting of multiple blocks was processed at a time. Multiple frames were used to calculate the average block computation times. The computation times for an average block of size NxN are presented in Table I for N of 32, 64 and 128 using CPU and GPU implementations. The total amount of blocks in a single frame depends on the block size. When the block size was 32x32, there were 2778 blocks. On sizes 64x64 and 128x128, there were 687 and 163 blocks respectively. The computations were done using a laptop computer with Core i7-3820QM processor and NVIDIA Quadro K2000M Graphics Card. Only one thread was used for CPU computations.

Radon transform was computed for 2N orientations. It was also computed using Matlab R2012A and built-in *radon.m* function as a reference for the CPU implementation. As the results in Table I show, the CUDA implementation is in total 18 times faster than CPU for 32x32 block size. For larger block sizes, the total speedup is 25 and 79 times for image blocks of size 64x64 and 128x128 respectively. The computation times for Radon transform and for total time are plotted as a function of block size in Fig. 7 and Fig. 8.

Table I
Computation timings for an average block

Block Size	Operation	Matlab (μs)	CPU (μs)	CUDA (μs)
32x32	Prepare		0.14	3.51
	Radon	777	111.11	4.14
	Wavelet		81.82	2.38
	Features		21.19	2.18
	Total		**215.44**	**12.11**
64x64	Prepare		1.26	11.73
	Radon	5872	1089.28	34.04
	Wavelet		319.38	7.59
	Features		71.14	6.24
	Total		**1470.58**	**59.63**
128x128	Prepare		16.82	43.50
	Radon	47301	20780.30	190.71
	Wavelet		1266.36	12.94
	Features		291.95	24.71
	Total		**21345.10**	**270.96**

In the small field test, four out of the five spruce trees were successfully found and cleaned using the automatized forest machine. The detection result of a example image is depicted in Fig. 9. Subfigure A shows the original test image and B shows the detection result computed for blocks of size 32x32 with 16 pixel overlap. Therefore, the texture and color classification is done on a grid of every 16th pixel. As there are 2778 image blocks computed for every frame, the total time for processing is 30.6 milliseconds. The classification time depends on the amount of training data. When the data consisted of 2000 features, the classification took 8 milliseconds per frame. Therefore, theoretically, the system can process 25 frames per second. However, our camera runs only at 15 fps and thus our level of algorithm optimization was sufficient. The speed of our system allows a lot more training data to be used. This can be an advantage as different lighting and environmental conditions are taken into account.

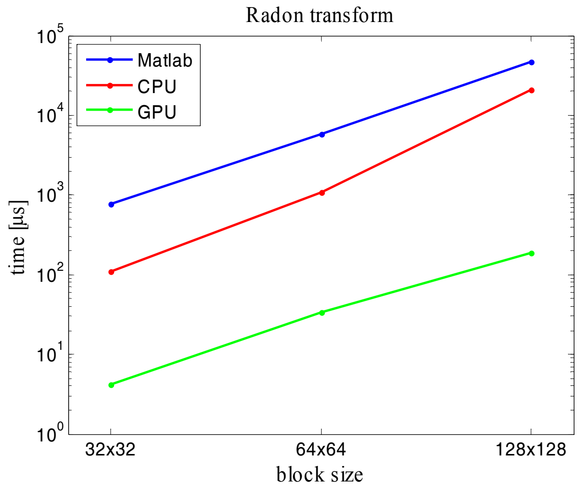

Fig. 7. Radon transform time using GPU, CPU and Matlab. The GPU implementation is many orders of magnitude faster than CPU and Matlab implementations.

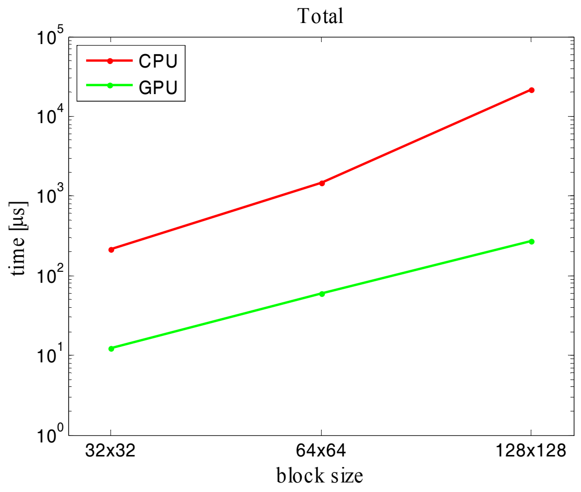

Fig. 8. Total computation time using GPU, and CPU. The total computation time includes all computation steps: Radon and wavelet transforms, feature extraction and k-NN classification.

The quality of spruce detection was tested using a data recorded from the field tests with five spruces. The data from other four spruces than the target tree shown in Fig. 9 was trained to the k-NN detector. The test and training data were from the same test and therefore had similar lighting conditions and weather. Therefore, the test results are slightly overoptimistic. The spruce tree detection in Fig. 9 is drawn using a blue circle. The tree is found at the center of a blob of large spruce detector votes. Spruce votes are depicted using orange dots. The size of a dot corresponds to the amount of spruce votes around the current image block.

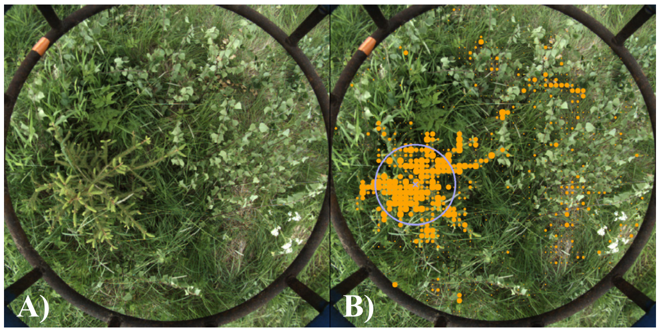

Fig. 9. The detection result for a typical spruce. A) The original image and B) the block-vice detection result plotted on the image. Larger orange dot means more spruce votes in the neighborhood of k-NN classifier. The blue circle represents an estimate for location of the spruce tree.

The quality used to estimate the detection reliability is plotted in Fig. 10. The decision threshold is drawn with a red dashed line. Areas, where the quality is below the estimated threshold, are not used to control the forest crane and are drawn using gray color in the figure. The Euclidean distance from the reference to the detected spruce was measured and it is plotted in Fig. 11. The reference data for spruce detection was generated by clicking the approximate center location of a spruce tree in every image separately. In the data, the tool is lifted over the spruce and the target is therefore moving from left to right during image sequence.

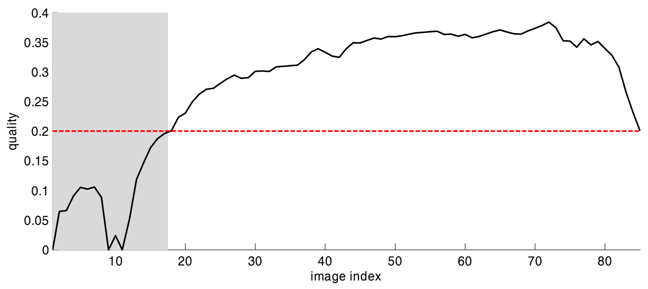

Fig. 10. The estimated spruce detection quality in sequential images as the tool is lifted over the young spruce tree. At the left and right edges of the figure, the tree is at the edge of the image. The gray area in the figure states that the quality is under the used threshold drawn using a red dashed line.

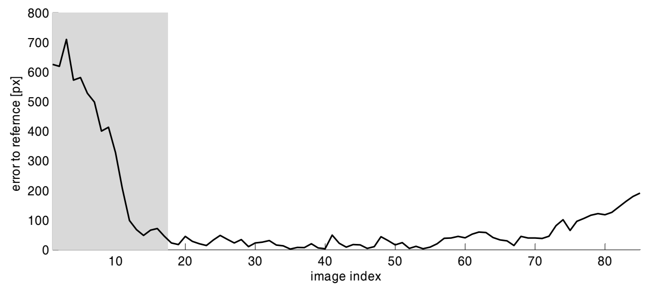

Fig. 11. The error to reference measurement in pixels as the tool is lifted over the young spruce tree. At the left and right edges of the figure, the tree is at the edge of the image, and the detection is not as reliable as in the center of the figure. Similarly as in Fig. 10, the gray area in the figure states that the quality is under the same determined threshold. The detection error is usually within 50 pixels.

As it can be seen from Fig. 11, the detection error is usually less than 50 pixels when the quality is high enough. With the large image indexes shown in the right part of Fig. 10 and Fig. 11, the tree is in the edge of the image and is seen only partly. The error increases as the target tree is going out of the view. This systematic error in the system is not crucial for the operation as the tree is controlled towards the center of the image. The robust accuracy of less than 50 pixels at the center areas of the image is enough to be successfully used in visual servoing the point cleaning tool over the target.

The Fig. 9 is the 34th image of the test result plotted in Fig. 10 and Fig. 11. In the image, most of the k-NN detector spruce detections are inside the blue circle drawn in Fig. 9B. Therefore, the quality plotted in Fig. 10 is quite high, as the quality is estimated by summing all spruce votes inside the circle and dividing it with a sum of all spruce votes in the image. The greediness of spruce detector can be adjusted using the detection threshold.

V. Discussion

The operational tests demonstrating the system were run on a small field with only five planted spruces and a set of deciduous trees and high grass complicating the task. The quality of spruce detection is therefore reported only briefly. We have focused to the real-time implementation of Radon and wavelet texture detection algorithm in visual servoing task in a complex environment. We have reported the speed gains which can be achieved using GPU computing. In addition to the fact that GPU is faster, the use of GPU computation frees CPU resources.

The quality of the spruce detection was tested against hand made reference measurements. In practice, the implemented spruce detection system was accurate enough to be used in visual servoing the forest crane and to point clean surrounds of the planted spruces on our test field. More advanced tests should be carried out in a real spruce plantation. Similarly, the developed user interface was mainly built for testing purposes, and more intuitive user interface should be developed. The future version could have a multipoint touch screen to make the interface for labeling the training data and controlling the crane easier. Similarly, the system could be tracking the operator and train new features automatically as the forest machine is operated manually.

A more detailed validation of the quality of spruce classification is still required. A more comprehensive data set has been collected for evaluating the quality of spruce detection algorithms and the GPU implementation. In this paper, we have used only Radon transform based texture detection methods. Different texture detection methods, for example Gabor filtering should be evaluated using the comprehensive data set. More detailed performance of our spruce detection algorithm will be published later.

Acknowledgment

The authors thank Mr Matti Öhman for developing the camera lift. We thank Mr Raimo Linkolehto for all his help in building the platform and arranging the tests. In addition we thank Mr Teemu Kemppainen for the manual reference measurement. The test locations were coordinated by MTT Agrifood Research Finland.

References

[1] J. Billingsley, A. Visala and M. Dunn, "Robotics in agriculture and forestry," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer-Verlag, 2008, pp. 1065.

[2] H. Hallongren and J. Rantala, "Commercialisation and International Market Potential of Finnish Silvicultural Machines," *Silva Fenn.,* vol. 46, pp. 583-593, 2012.

[3] S. Härkönen, "Effects of silvicultural cleaning in mixed pine deciduous stands on moose damage to scots pine (Pinus sylvestris)," *Scand. J. for. Res.,* vol. 13, pp. 429-436, 01/01; 2013/02, 1998.

[4] K. Uotila, J. Rantala and T. Saksa, "Estimating the Need for Early Cleaning in Norway Spruce Plantations in Finland," *Silva Fenn.,* vol. 46, pp. 683-693, 2012.

[5] A. Karlsson, A. Albrektson, B. Elfving and C. Fries, "Development of Pinus sylvestris Main Stems Following Three Different Precommercial Thinning Methods in a Mixed Stand," *Scand. J. for. Res.,* vol. 17, pp. 256-262, 01/01; 2013/02, 2002.

[6] K. Jafari-Khouzani and H. Soltanian-Zadeh, "Rotation-invariant multiresolution texture analysis using Radon and wavelet transforms," *Image Processing, IEEE Transactions on,* vol. 14, pp. 783-795, 2005.

[7] K. Jafari-Khouzani and H. Soltanian-Zadeh, "Radon transform orientation estimation for rotation invariant texture analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 27, pp. 1004-1008, 2005.

[8] M. R. Hejazi, G. Shevlyakov and Yo-Sung Ho, "Modified discrete radon transforms and their application to rotation-invariant image analysis," in *Multimedia Signal Processing, 2006 IEEE 8th Workshop on,* 2006, pp. 429-434.

[9] J. Kalmari, T. Pihlajamäki, H. Hyyti, M. Luomaranta and A. Visala, "ISO 11783 compliant forest crane as a platform for automatic control," in *Agricontrol 2013,* Espoo, Finland, 2013.

[10] M. Brady, "A Fast Discrete Approximation Algorithm for the Radon Transform," *SIAM J. Comput.,* vol. 27, pp. 107-119, 02/01; 2013/02, 1998.

[11] Jie Liang and T. W. Parks, "A translation-invariant wavelet representation algorithm with applications," *Signal Processing, IEEE Transactions on,* vol. 44, pp. 225-232, 1996.

[12] Y. Ohta, T. Kanade and T. Sakai, "Color information for region segmentation," *Computer Graphics and Image Processing,* vol. 13, pp. 222-241, 7, 1980.

[13] B. L. Steward and L. F. Tian, "Real-time weed detection in outdoor field conditions," pp. 266-278, January 14, 1999.

[14] E. Gose, R. Johnsonbaugh and S. Jost, *Pattern Recognition and Image Analysis.* Prentice Hall PTR, 1996.

[15] Nvidia, "CUDA C Programming Guide Version 4.2," *NVIDIA: Santa Clara, CA,* 2012.