MVA2009 IAPR Conference on Machine Vision Applications, May 20-22, 2009, Yokohama, JAPAN

Image Processing Architecture for Real-Time Micro- and Nanohandling Applications

Tim Wortmann[1]*, Christian Dahmen[1], Robert Tunnell[1] and Sergej Fatikow[1]

[1]Division Microrobotics and Control Engineering, University of Oldenburg, Germany

Abstract

This article presents a software architecture for image processing applications. Its intended use is in the automation of manipulation processes at the micro- and nanoscale. The main requirements for this architecture are the capability for real-time processing, the flexibility to cover a wide range of different applications and simplicity of usage. Some of the biggest challenges include online changes of control- and data-flow, integration of a large number of devices for microscopy and the need for high-speed result forwarding. The architecture has been implemented and tested extensively. Three key applications are presented: object tracking, classification of biological cells and 3D image reconstruction for scanning electron microscopes.

1 Introduction

Over the last decades, micro- and nanotechnology developed from a discipline of fundamental research to an indispensable sector of industry [6]. While many individual production steps are still carried out manually, much effort is spent to provide automated procedures for micro- and nanotechnological applications [2]. Some of the most important fields of research are the manipulation of biological cells and the assembly of nanoelectronic devices.

In order to carry out processes on the micro- and nanoscale, visual feedback has to be used. For automation purposes, image processing algorithms are utilized to obtain information on the current state of the manipulated objects. Up to now, the algorithms used in nano- and micromanipulation are often implemented in separate monolithic software packages. This has the disadvantage that changing application settings with different algorithm requirements demand reimplementation of parts of the software or a change of the software. To prevent this, a graph-based scheme has been implemented to enable flexible usage of any implemented algorithm in different manipulation setups.

The new image processing architecture will be proposed in section 2. We will demonstrate its capabilities by presenting three sample applications that have been implemented and tested: object tracking using active contours, cell classification using shape features and 3D image reconstruction for scanning electron microscopes (SEMs).

*This work was supported in part by the European Community: Project HYDROMEL (026622-2). Corresponding Author: `wortmann@informatik.uni-oldenburg.de`

2 Architecture

In order to enable flexible image processing, it is necessary to introduce an architecture which makes this flexibility possible. Principally, most image processing tasks are easily described as an input-processing-output (IPO) model. To increase the possibilities, this can be detailed into pipeline structures. If this is further generalized, graphs can cover most image processing tasks. The proposed architecture is based on a graph structure to enable flexible data flow and complex image processing applications. The graph nodes can be exchanged, added, modified and removed during runtime. The key components of the architecture shown in figure 1 will be described in the following.

The **graph** manages the nodes and connections. Nodes are created in a node factory and connected to each other using ports.

The **graph nodes** are implemented as threads, leading to a certain level of coarse parallelism. All nodes have the same interface while fulfilling different functions. Input nodes function as data sources, output nodes are data sinks. Processing nodes implement algorithms working on the image data. Sync nodes enable parallel asynchronous data streams. All nodes have parameters which control their behaviour.

Each node contains **parameters** that are implemented to be of arbitrary number and can have different types. All parameters can be set during runtime and may be used to set up the node function.

All nodes have **ports** for communications. There are two types of ports for input and output. Input nodes do not have input ports, while output nodes lack output ports. Each output port can connect to an arbitrary number of input ports. By this, the parallel execution of different algorithms is inherently enabled. All ports are typed, and it is possible to specify input ports as optional or required.

The **input nodes** implement the interface to the imaging hardware, which can be USB, Ethernet or Firewire cameras, video file input, specific framegrabbers or interfaces to SEMs. The whole functionality is encapsulated in the node so that a standardized interface is available for any image source. The input nodes has functions such as continuous or triggered capturing and parameters such as image size and frame rate.

The **output nodes** realize sinks for the data generated in the graph. Possibilities include video file output, screen display, data file output for non-image data, network streaming and interfaces to other software packages. Again, all functionality is encapsulated, so that all output nodes can be used interchangeably.

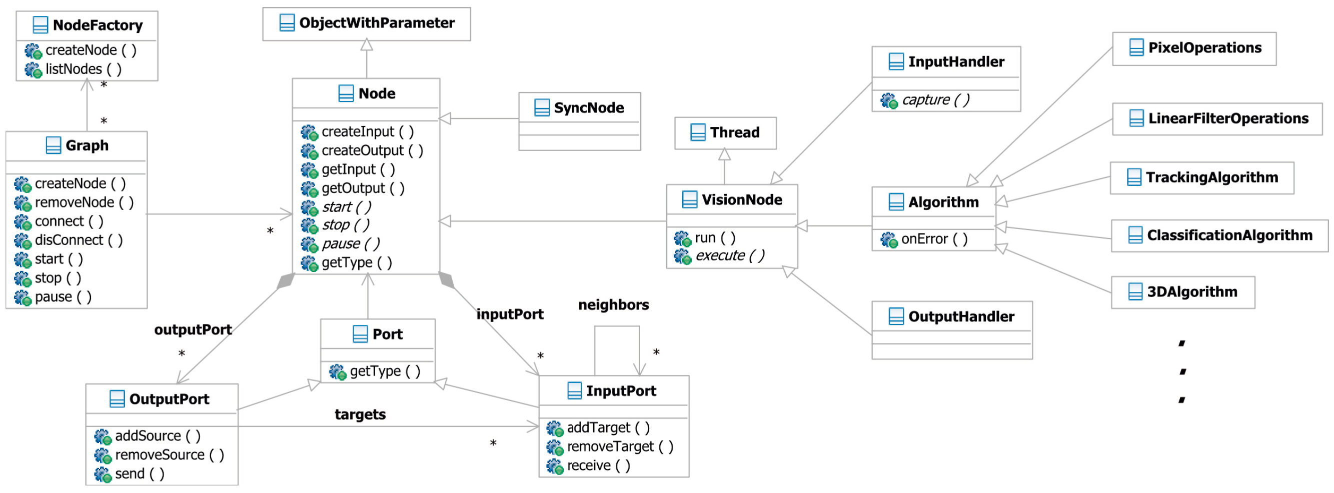

Figure 1: Class diagram of the image processing software core, implementing a graph structure. Edges of the graph are implicit by setting the target port of output ports.

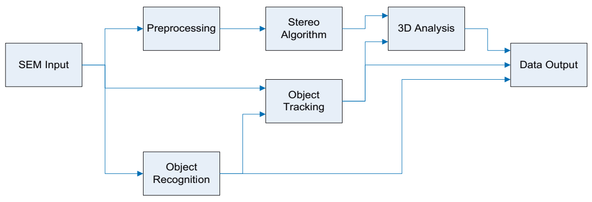

Figure 2: Example image processing graph for a manipulation setup. Object recognition initializes object tracking, object tracking determines object position and delivers position information to output and to 3D analysis algorithm. The stereo algorithm generates a depthmap which is evaluated at the position delivered by the tracking.

The **algorithm nodes** are used to implement arbitrary image processing algorithms. The range of algorithms covered extends from simple point operations to complex and computationally expensive algorithms such as depth from stereo or different tracking or recognition algorithms. The key aspect is that the interface for these algorithms is similar, while the exact parameters vary. This allows the user to exchange certain algorithms with other algorithms carrying out the same function during runtime if needed.

With this graph concept, also more complex setups can be realized. An example is shown in figure 2. The algorithms in this case depend on each other. The recognition algorithm is used to initialize the tracking algorithm, after which it is paused for the subsequent frames. The tracking algorithm extracts the two-dimensional position of an object from the images, delivering this position to the output node and to the 3D analysis node which extracts the z-position from the depth map delivered by the stereo algorithm. The requirements on the used hardware mainly depend on the used algorithms. A simple graph has very little overhead apart from capturing and display.

All components of the software can be controlled either by the graphical user interface, or remotely via network. For the setup of graphs, different functions are provided. All operations with nodes can be executed: insertion, removal, connection of two nodes. Additionally, each node may be halted and started independently. All parameters of the nodes can deliver valid ranges for their values and can be set remotely. Operations such as image acquisition and certain algorithms can be triggered via the GUI or remotely. Combining this with the network output node, a controller has full access to all aspects of the image processing software.

For automation scenarios, different image processing algorithms are used. In the following, three different methods are presented which have been successfully integrated.

3 Object tracking

One problem for manipulation processes on the nanoscale is the difficulty of estimating the position of objects during handling. The position may differ due to e.g. thermal drift or backlash. To solve this problem, object tracking algorithms are used with image sources such as microscope cameras and SEMs.

Different approaches have been proposed for object tracking on the nanoscale. These are based either on template matching [8], rigid body models [5] or active contours [9]. The algorithm here is inspired by the active contour approach (see also [7]), which is extended to deliver estimates for out of focus displacement by analyzing object focus.

In order to gain robustness against noise, the active contours are not used in their original formulation, but using an energy function derived from region statistics. By this, not only the contour outline, but the whole area enclosed by the contour contributes to the minimization process. This ensures higher stability of the contour.

Additionally, by evaluating the area statistics, it is possible to derive a sharpness measure for the object. When the maximal sharpness and the actual sharpness of the object is known, an estimate for object displacement can be calculated. If this is combined with the two-dimensional position information, a three-dimensional object position is obtained. The

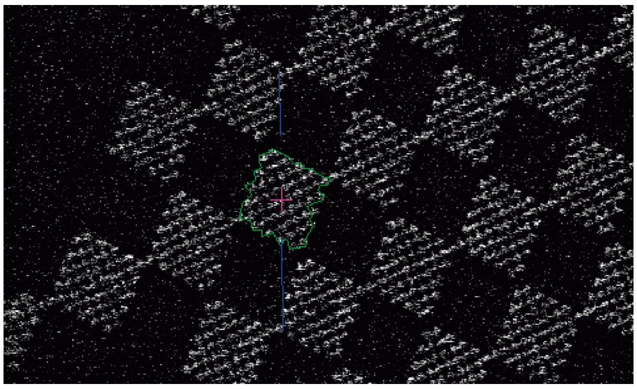
Figure 3: Active contour based tracking algorithm applied to part of a chessboard calibration pattern in the SEM.

two-dimensional part of the algorithm is shown in figure 3.

Important for this algorithm is that the execution should be possible in real-time, which posed no problems for the developed architecture. Also, the algorithm in addition to the images depends on information about the actual working distance of the SEM, which was easy to deliver using the SEM input node. The active contour segmentation details can be transferred to subsequent algorithms for further analysis of the enclosed object.

4 Cell Classification

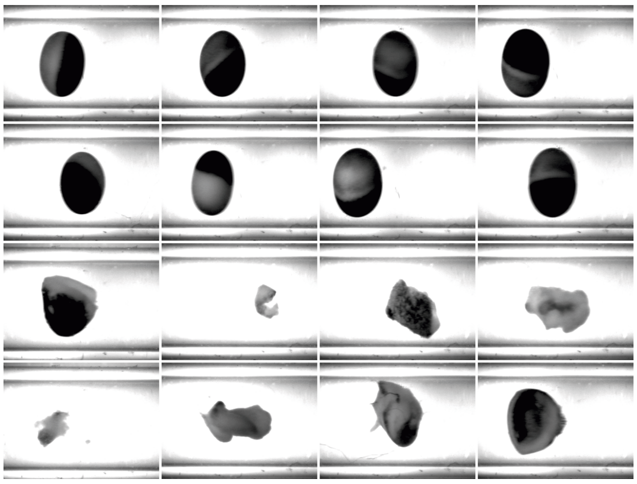
Figure 4: Selection of image scenes showing viable (upper two rows) and damaged (lower two rows) cells in motion.

Another application which has been implemented using the new architecture is the classification of biological cells. We use oocytes of Xenopus Laevis (African clawed frog) intended for later microinjection. During separation, cells can be damaged and need to be removed from the viable cells. The cells flow through a microfluidic channel and are classified on the fly. This process is often referred to as image flow cytometry [1]. Figure 4 shows a selection of cells and particles in motion.

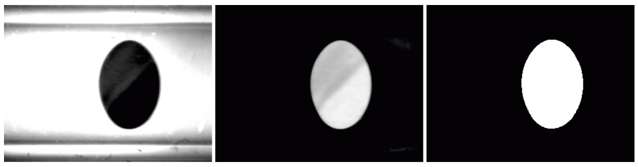
Figure 5: Image sequence depicting the segmentation procedure of a cell image scene using background subtraction.

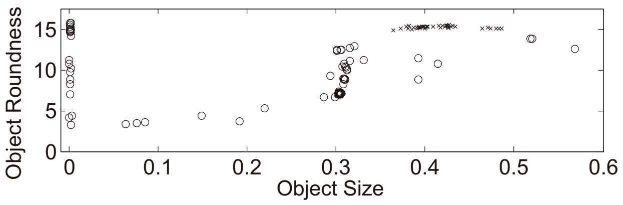

Figure 6: Scatter plot of relative object roundness and object size. Circles mark damaged cells and particles, viable cells are marked by a cross.

Defects in the outer cell hull can be detected by using a shape descriptor. Initially, the cell is segmented from the background using background subtraction (see figure 5). We choose two features of the object shape:

1. Size approximated by the number of pixels
2. Object roundness R as described in [3]:

$$R = \frac{4\pi \cdot Area}{Perimeter^2} \ . \tag{1}$$

Figure 6 shows a scatter plot of the two shape features obtained by experiment. The object size is normed to half the size of the region of interest, R is outweighted by a factor of 18. A non linear Support Vector Machine [10] has been trained on these data points. Reliable defect detection was achieved at up to 60 fps.

5 3D SEM

Many micro- and nanorobotic applications, e.g. pick and place operations require z positions of objects in addition to the x and y positions. The SEM normally delivers two-dimensional pictures and therefore a stereoscopic lens system is used to acquire two separate images with a slightly different viewing angle. The aim of the 3D module is to locate corresponding points in each stereo image and calculate the disparity between them to determine the z position. The new software architecture described in section 2 provides a solid basis to integrate an already existing, standalone version of a stereoscopic algorithm. Thereby, a complete image processing pipeline including 3D image acquisition, calculation and analysis can be built. Disparity calculation of the 3D module [4] consists of two parts: The estimation layer, which calculates a number of possible disparities per image pixel and the coherence layer, which makes a coherence analysis to calculate an unique result. Many applications in the SEM e.g. the manipulation of carbon nano tubes (CNTs), require a flexible embedding of image preprocessing because of the SEM images' inconsistent quality. Due to the new pipeline-based architecture, the image acquisition parameters and the preprocessing can be changed with the opportunity to directly observe the outcome. The figure 2 shows a possible variation of an image processing pipeline using the 3D module.

Because of the integration, the 3D module is now capable to run from the acquisition to the analysis as a well defined, stable and coherent image processing unit. To fit the real-time requirements of many applications, it is necessary to optimize the runtime of existing 3D module. The main approach is to parallelize the calculation of the disparity values. The parallelization mainly effects the estimation (figure 7) and coherence layer of the stereo algorithm. These changes result in a speed-up of up to 300% depending on the image size. As a validation example, the 3D

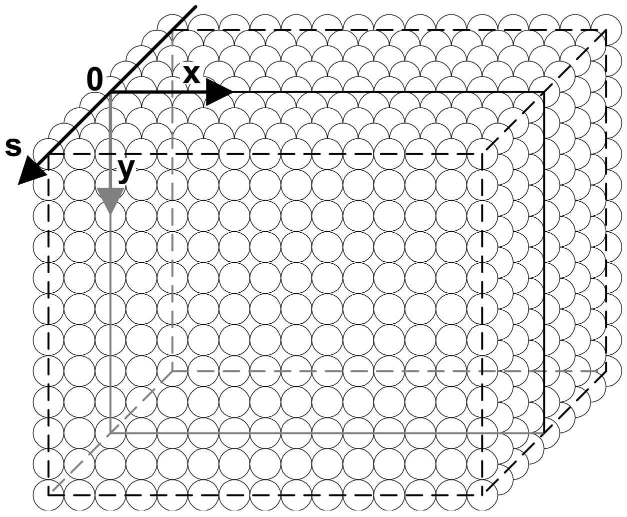

Figure 7: Estimation cube. Each of the cube elements is a possible disparity and can be independently calculated.

calculation of a CNT and a electro-thermally actuated microgripper (figure 8) performed in an image with 200 x 200 pixels. Before optimization, the algorithm needed 2.6 s for the calculation and analysis of the disparity values. The parallelized version solves the same task in 1.15 s. Additionally, an extra acquisition module was implemented to load stereo images from disk and enable the offline calibration of the 3D module. Thanks to the new architecture, different optimization implementations can be benchmarked and validity as well as accuracy of the results can be checked at runtime.

6 Summary & Outlook

In this work, a flexible image processing architecture has been presented which enables the automation of micro- and nanohandling processes. The architecture has been implemented and integrated into a pre-existing distributed control architecture. A comprehensive test series proved real-time capability, a high level of parallelism and efficient usage of resources in

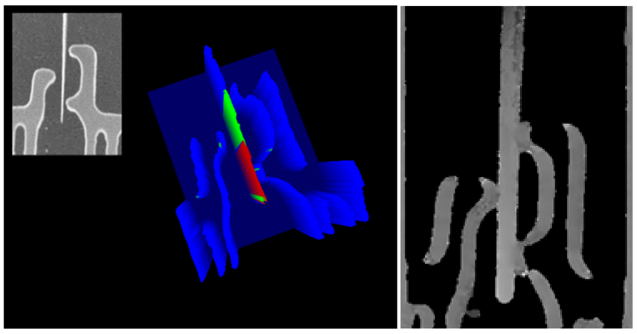

Figure 8: 3D visualization of the validation example (left) and Filtered disparity map (right) of a CNT manipulation with a electro-thermally actuated microgripper.

several manipulation setups. Three image processing applications have been explained in more detail. Overall the software architecture is found to be flexible enough and usable for automation purposes.

Additional steps will be taken to enable further parallelization by using commodity graphics processors (GPGPU). First attempts have shown that this approach has high potential to increase the real-time capability of certain very complex algorithms. Extensions to our software architecture will be developed to synchronize GPGPU and main processor calculations and to enable a certain amount of load balancing.

References

[1] L. Bonetta. Flow cytometry smaller and better. *Nature methods*, 2005.

[2] S. Fatikow (Ed.). *Automated Nanohandling by Microrobots.* Springer Series in Advanced Manufacturing. Springer, 2008.

[3] M. L. Hentschel and N. W. Page. Selection of descriptors for particle shape characterization. *Particle & Particle Systems Characterization*, 2002.

[4] M. Jähnisch. *3D-image system for nano handling in the scanning electron microscope.* PhD thesis, Universität Oldenburg , Department für Informatik, 2008.

[5] B. Kratochvil, L. Dong, and B. Nelson. Real-time rigid-body visual tracking in a scanning electron microscope. In *Proc. of the 7th IEEE Conf. on Nanotechnology (IEEE-NANO2007), Hong Kong, China*, August 2007.

[6] B. Nelson, J. Abbott, Z. Nagy, and F. Beyeler. Robotics in the small, part i and ii. *IEEE Robotics and Automation Magazine*, 14:92–103, 111–121, 2007.

[7] T. Sievers. *Echtzeit-Objektverfolgung im Rasterelektronenmikroskop.* PhD thesis, University of Oldenburg, 2007.

[8] T. Sievers and S. Fatikow. Visual servoing of a mobile microrobot inside a scanning electron microscope. In *Proc. of IEEE Int. Conference of Intelligent Robots and Systems (IROS), Edmonton, Canada*, 2005.

[9] T. Sievers and S. Fatikow. Real-time object tracking for the robot-based nanohandling in a scanning electron microscope. *Journal of Micromechatronics - Special Issue on Micro/Nanohandling*, 3(3-4):267–284(18), 2006.

[10] I. Steinwart and A. Christmann. *Support Vector Machines.* Information Science and Statistics. Springer, 2008.