# A Conceptual Structure for Computer Vision

Gregor Miller, Sidney Fels and Steve Oldridge
*Human Communication Technologies Laboratory*
*ECE, University of British Columbia*
*Vancouver, Canada*
{*gregor, ssfels, steveo*}*@ece.ubc.ca*

***Abstract*—The research presented in this paper represents several novel conceptual contributions to the computer vision literature. In this position paper, our goal is to define the scope of computer vision analysis and discuss a new categorisation of the computer vision problem. We first provide a novel decomposition of computer vision into base components which we term the *axioms of vision*. These are used to define researcher-level and developer-level access to vision algorithms, in a way which does not require expert knowledge of computer vision. We discuss a new line of thought for computer vision by basing analyses on descriptions of the problem instead of in terms of algorithms. From this an abstraction can be developed to provide a layer above algorithmic details. This is extended to the idea of a formal description language which may be automatically interpreted thus allowing those not familiar with computer vision techniques to utilise sophisticated methods.**

***Keywords*-Computer Vision; Vision Systems; Vision Development;**

## I. Introduction

This paper represents several novel conceptual contributions to the computer vision literature as a basis for discussion of computer vision system development. In this position paper, we apply a software-engineering based approach to provide a novel decomposition of computer vision into a basis which we term the *axioms of vision*. We also discuss a new line of thought for computer vision by basing analyses on descriptions of the problem instead of algorithmic details. This is extended to the novel idea of a formal description language which may be automatically interpreted thus allowing those not familiar with computer vision techniques to utilise sophisticated methods.

Our goal is to define the scope of computer vision analysis and discuss a new approach to categorisation of vision and development of new algorithms. The ideas presented will hopefully lead to new forums for vision researchers to share ideas, algorithms and software, which will in turn help provide access to computer vision tools to those not familiar with the techniques themselves. Ideally this would lead to increased industry involvement with the academic computer vision community and industry use of vision in products.

Much of our motivation comes from the success of similar endeavors in computer graphics. Frameworks such as OpenGL and DirectX provide an abstraction which offers the use of advanced graphics techniques to a large user base, not just specialists in graphics. The abstraction gives programmers access to conceptual processes (e.g. drawing a textured triangle) but hides the complexity of the operation from the programmer. However, graphics is a structured problem and is more easily represented in concept. The image understanding problem is unstructured and requires a higher level of sophistication to extract meaningful models. Our discussion in this paper attempts to bring control of sophisticated concepts such as image registration to a level similar to that of drawing a triangle.

The central theme of our contributions are *accessibility*: we would like to see computer vision becoming more accessible to researchers and developers alike. With this in mind we discuss a reversal in the prevalent form of thinking within the community, and extend the taxonomy and decomposition of the vision problem into the beginnings of a description model which may be used by those not deeply familiar with vision techniques.

Widespread adoption of computer vision techniques is inhibited by the lack of a framework that provides an abstraction over algorithmic detail. For example, a developer wishing to use face detection for a system should not need to know about Haar Descriptors (as is required when using OpenCV), or any other specific algorithm. Researchers should also not necessarily need to know the specifics if their research utilises the result of detection methods. If the concept is understood and the problem being dealt with can be described sufficiently well then an interpreter within a suitable framework should be able to infer from the description how to solve the problem. We propose two levels of abstraction: one for researchers (Section IV-A) which provides the flexibility required to develop new algorithms and integrate existing ones, and one for developers (Section IV-B) which provides a problem-centric view of vision where specific algorithms are hidden.

There have been many attempts to create open repositories of software supporting the vision community [1], [2], [3], [4], however expert vision knowledge is required, such as which algorithm can be applied to solve a particular problem. There is usually also a combination of image capture, convenience utilities (such as image I/O) and specific algorithms for image processing and understanding contained within these libraries. This combination of

features demonstrates that these frameworks suffer from a lack of sufficient conceptual organization of the vision problem's constituent tasks. Previous research has shown that lack of scope definition and overlap across frameworks leads to a breakdown in component reusability [5]. Therefore we propose the following classification of scope for computer vision:

| | | |
|---|---|---|
| **Access** | : | Retrieval of image data |
| **Transfer** | : | Communication of image data |
| **Convert** | : | Conversion into required format |
| **Modify** | : | Applying filters, crop, transforms, etc. |
| **Analyse** | : | Using vision to understand a scene |

Decomposing the problem in this way promotes code re-use as well as focussing development effort on well-defined parts of computer vision. The other components in the computer vision pipeline have various example solutions, such as Quicktime7™ for access, Hive [6] for transfer, ImageMagick for conversion and CoreImage™ for modification. As yet there are no known solutions to the **Analysis** problem which limit their scope to analysis while also being comprehensive and accessible to those not specialised in computer vision techniques. We aim to outline a methodology which may help the community accomplish this goal.

In this paper we plan to recursively apply the same idea of classification and decompose the **Analysis** component into a basis we call the axioms of vision, discussed in Section III. Then by applying a bottom-up approach we can compose algorithms from the axioms (Section IV-A), which becomes the researcher-level framework for abstraction and algorithm integration.

One of the main contributions of this work is a reversal in the way we present our thoughts on vision. There are many occurrences where work such as surveys and books categorise based on algorithms. We believe this should be reversed, and that taxonomies should be based on problem conditions and not algorithm suitabilities. For example, if one were to look up "Tracking" in the index of "Computer Vision: A Modern Approach" by Forsyth and Ponce [7], there are many listings of various algorithms and applications. If this were reversed, there would be a chapter devoted to tracking which describes the problem and the various conditions under which the problem appears, and then a suitable set of algorithms are given which can solve the tracking problem under certain conditions.

## II. Related Work

Makarenko [5] provided justification for confining scope to smaller areas which can be readily encapsulated, thereby promoting reuse of code. This then creates a general basis for development and sharing of algorithms, systems and other software with well-defined interfaces. We wish to see the development of a vision system which is analogous to the boost C++ libraries - a peer-reviewed SDK which contains proper encapsulation and remains flexible enough to be used by both researchers and developers.

The previously published literature in this area is generally similar to textbooks such as Forsyth and Ponce [7], which provide categorisations of computer vision. These are generally algorithm-centric in their organisation, and as such are a resource mainly for computer vision researchers. We suggest that the categorisation should be problem-centric (e.g. "Tracking", "Reconstruction") instead, allowing those inexperienced in computer vision to survey the available methods of solving that problem.

Surveys also provide taxonomies and categorisations of individual areas within vision, but from our experience none survey the material from a problem-centric perspective. The following are a sample from various fields within vision. Zhang et al. [8] present a survey of unsupervised segmentation techniques: from the title it is clear that algorithmic detail influences the paper; Zitova and Flusser [9] present a survey of image registration, dividing their survey into how algorithms perform feature detection, feature matching, mapping function design and image transformation and re-sampling, which are all specific to the algorithms used to implement registration. Scharstein and Szeliski [10] present a taxonomy of dense stereo correspondence based on matching cost computation, cost aggregation, disparity computation and optimisation, and finally disparity refinement. Again, these are algorithmic details and do not focus on the conditions under which the algorithms would work. Campbell and Flynn [11] present a survey of object representation and recognition methods, categorised by items such as the appearance measures used, representation such as silhouettes or 3D model, or from the data sources such as images or range data. This is closer to the idea we describe here, in that they use the data sources for part of the taxonomy and that forms part of the original problem; however the algorithms are categorised by their implementation specific detail. We contend that surveys should also provide a problem-based context so that it is clear under what conditions the algorithms perform and which ones may be used to solve particular problems.

The other source of previous work for our idea lies in development and the software-engineering approaches applied to vision libraries. While OpenCV [1] provides many of the components it does not separate out the different elements into thin layers of abstraction. For example, useful functions such as CornerHarris, CalcHist, Filter2D, Sobel and so on require considerable knowledge about how to solve a particular image processing problem, however, they offer little insight into what the problem is, making it difficult to adjust when the problem changes, which presents challenges for reuse and scalability in real-world contexts. We see the axioms of vision as an abstraction that builds on top of algorithmic approaches such as OpenCV to leverage

the specialized knowledge contained in their impressive array of algorithms. Thus implementations of the axioms may use OpenCV effectively and allow non-specialists to take advantage of its functionality. Likewise, other libraries that provide vision algorithms can be used to implement components of a framework based on the axioms such as itk [12], ImageJ [13], Gandalf [2], Camellia [4], and VXL [3]. These libraries similarly do not define the scope of problems and are not accessible to general developers.

The advantage of our methodology is a problem-based index, i.e. it is possible to look up "Reconstruction" and find either a list of suitable algorithms (for researchers) or a known solution based on a software library which supports our decomposition of the computer vision problem. This is the reverse of the normal algorithm-centric approach of defining an algorithm such as optic flow and then listing the applications of this algorithm. In the following sections we decompose the vision problem into sub-problems called axioms and then discuss the composition of algorithms from these axioms and how a problem-centric system could be developed to allow non-specialist programmers to utilise vision techniques.

## III. The Axioms of Vision

The axioms of vision are presented here as individual components which each perform a subset of tasks which can be sensibly categorised together. We adopt the term *axiom* from the mathematical usage, as we wish to use these as building blocks from which to derive new constructs. The advantage of this approach is to abstract representation, type and algorithm detail within these components so that they can be combined together (connectivity) and built upon to create new algorithms which can also be part of a higher-level component to provide an abstraction over the algorithm. This allows the same framework to be used by both vision researchers who require details of the representation and algorithms and application developers who require the use of vision techniques who only need to have knowledge of the problem they want to solve.

We apply a bottom-up approach for the composition of vision algorithms, which requires us to define the bottom layer. The axiom of vision could be defined as low-level mathematical representations, such as an M×N matrix (which could represent images, feature vectors, etc.), but this level is not capable of encapsulating concepts. They could also be defined as high-level problem types such as "Tracking", "Reconstruction" etc., but these do not offer much flexibility or reuse of existing components. Our target is to define the axioms such that they are low-level enough to represent the majority of vision problems when combined but high-level enough to encapsulate core concepts of vision. Our process is analogous to defining a specification for a software engineered system using an object-oriented basis.

The axioms share common elements which would be defined globally, such as images and other basic data types. Colours, images and time are all treated as continuous signals. For example colour could be represented as RGB with each channel represented in the interval $[0, 1]$. This provides for high dynamic range, relighting, or various processing methods where discrete sampling is not sufficient. Image width and height would be represented similarly, with an additional value for aspect ratio. This would allow for an abstraction of image resolution (based on the premise that the sensor size and lens dictate the scale, the sensor resolution does not). Treating time as a continuous signal allows different inputs to a system following this model: different frame-rate cameras, or getting interpolated results from the system instead of at the same discretised moments of the input space. The discretisation of space and time is considered at a lower level, hidden from the users.

Each axiom should define its own types if the global types are not sufficient. Each type should be defined at most once, within the axiom that should sensibly encapsulate the associated concept. For example, an intrinsics calibration structure could be contained within the *Camera* axiom (assuming this is not represented as a matrix, but separately). The extrinsics is a matrix transformation, and so would use the type from the *Transform* axiom.

We do not claim the following is an exhaustive list of possible axioms for vision based in a problem-centric taxonomy, however these are sufficient to solve various example problems, as we demonstrate in Section IV-B. Each axiom can be divided into a description of its tasks and representations, and the processing it would perform as an implementation. We term the implementation of an axiom a *unit*.

### A. Mathematical Axioms

Mathematical axioms encapsulate the functionality required to perform transformations, optimisations and various other mathematical formulations in computer vision.

**Transform**: Essentially this contains the mathematical descriptions necessary for transformations of vision objects. This would include linear algebra, geometry and other formulations necessary for computer vision tasks. The representation of mathematical types for transformation is also defined here.
The *Transform Unit* would perform all mathematical transformations under the definition, for example matrix multiplication and projections.

**Optimise**: Various methods of performing optimisations are described for use by researchers for application with other axioms, such as dynamic programming, linear programming, greedy methods, graph cuts etc.
The *Optimise Unit* would provide type-independent or

generalised implementations of the various methods, useable in conjunction with data produced by other axioms.

**Similarity**: One of the central themes of vision is establishing similarity, used for correspondence, recognition, tracking, etc. This axiom describes various metrics for evaluating similarity, based on the different axioms defined below.
The *Similarity Unit* would provide the functionality of these metrics, for the various types defined in other axioms (e.g. object description).

### B. Source Axioms

Encapsulation of source information is accomplished with the following set of axioms. These are intended to describe acquired data or the source of acquired data, such as cameras and images, and the properties of images, such as noise and light.

**Camera**: The Camera axiom provides a description of a general camera. This could include various models (pinhole, thick lens, etc.) as well as specific parameters for calibration (principal point, focal length, etc.). The description could also take into account various metadata formats such as EXIF for digital imaging.
The *Camera Unit* is a description of the capture device used, and so it would not provide any processing methods. However, it would return the various matrices and other mathematical representations for cameras for use by the *Transform Unit*.

**Image**: This axiom provides descriptions of low-level image processing techniques and image representations, such as a Canny filter or resize operation.
The *Image Unit* would perform the image processing techniques described, accepting an image and producing a filtered image.

**Noise**: A description of the noise within data or noise to add to data.
The *Noise Unit* would provide noise removal, reduction and addition methods for various data types, such as images.

**Light**: A set of lighting models which can be used to approximate the lighting observed in a scene. This could follow simple systems such as those in computer graphics, or be a simple description of intensity variation across images (such as those with and without flash, or a set of images to be combined into one high dynamic range image).
The *Light Unit*'s responsibilities are vague as some could be carried out by the *Image Unit*; however various lighting descriptions could lead to filters for comparing lighting in an image or for providing illumination invariant metrics.

**Focus**: A set of descriptions of focus such as how to establish in-focus regions or compare focus levels between image patches. This could be used for depth-from-defocus, focal stacking or simply for establishing focus in images within applications which have control of a camera lens. Lens parameters etc. would be contained within the Camera axiom.
The *Focus Unit* would contain descriptors and metrics to, for example, evaluate focus or depth-from-defocus.

**Blur**: The Blur axiom is distinguished from the Focus axiom due to the deliberate nature of focus: blur implies a motion, either on the part of the camera or the subject, whereas focus is deliberately used for creative effect or for measurement (such as depth from focus). This axiom contains descriptions of blur, either as a kernel, a motion, or a higher-level indication of the type of blur.
The *Blur Unit* could contain items such as a blur-invariant feature descriptor [14], [15] or methods for producing blur in images.

### C. Model Axioms

The idea of the model axioms is to represent abstract concepts used within computer vision in an accessible way to users of the framework. Each of these is fundamental to computer vision and a clear, concise representation and interface would make it much simpler to combine them together to solve vision problems.

**Model**: The Model axiom contains descriptions of models used in vision: geometric, probabilistic or example-based.
The *Model Unit* would not need to provide much in the way of processing, except perhaps conversion from example-based to an internal format (which could be geometric or probabilistic).

**Object**: This is a complicated axiom, since it contains the description of objects. The concept of an 'object' in vision is vague, and so a description would need to be sufficiently flexible or a concrete definition supplied. The goal is to have the description which allows tasks such as object recognition to be accomplished.
The *Object Unit* would provide types and descriptions of objects. Recognition would likely be performed through a combination of Optimisation, Matching and Similarity.

**Match**: The idea of matching is comparable to Similarity, however it is a separate axiom to highlight the difference in *what* it is that can be matched. Similarity provide the metrics for comparison, the Match axiom uses these metrics under different applications, such as image windows,

patches or regions, bilinear interpolation, geometric shapes, and so on. It would also deal with scaling issues.
The *Match Unit* would implement these approaches, such as matching image regions [16] which could be used in stereo matching, or matching descriptions against image patches for object recognition.

**Move**: This axiom offers descriptions of motion, i.e. a sophisticated dynamics model which could be used for tracking.
The implementation as the *Move Unit* could accept previous estimates as input and provide a prediction as output. The concept of keeping state is left out and would form part of a tracking system (such as using the Optimisation axiom for dynamic programming).

**Colour**: This is one of the seemingly simple axioms that actually hides a more complicated issue - what is colour? If someone refers to 'red', what is it exactly that they mean? This axiom provides representations of colour which correspond to solid definitions. Initially our thoughts are to use probability density functions for each colour channel of an image point sample, which allows the concepts of 'red' as a specific colour e.g. (1,0,0) or as a spread of colour around the specific colour sample.
The *Colour Unit* would perform conversions between colour formats and provide mathematical operations on colour. The description combined with these operations should hide the internal format of colour from users, while still allowing specific colours or colour ranges to be represented.

**Depth**: This axiom provides description and types related to depth, such as depth maps from a specific point (projective) or from a plane (orthographic). Depth maps can be constructed from correspondence maps or defocus structures.
The *Depth Unit* would construct depth maps based on the output from other axioms, as well as provide depth-based processing (grey-level image maps of depth, 3D point maps etc.).

**Matte**: To perform background subtraction or foreground extraction, descriptions of the background or foreground are required. This could be a simple chroma-key approach (and so a colour range could be defined as a PDF using the Colour axiom), a model (geometric, appearance) or a background example image.
The *Matte Unit* would perform background subtraction or foreground extraction based on the description, and provide a mask over the original image.

**Material**: This describes material properties of regions within an image which may help in understanding of the scene, such as translucency or reflectivity.
The *Material Unit* would not extract these, the axiom provides the description which may be used by higher-level components.

**Shape**: This axiom describes the shape of coherent regions within the scene. This could be geometric with primitives or more advanced descriptors[17], or an example-based system.
The *Shape Unit* would provide mechanisms to find similarly shaped regions within an image, in conjunction with the Matching and Similarity components.

**Texture**: The texture of a scene is described here, which may then be used for image decomposition, analysis or synthesis.
The *Texture Unit* could perform analysis and synthesis using the texture descriptor, as well as finding regions of similar texture with the Matching and Similarity axioms.

**Appearance**: This describes the appearance of some region, which may be an object, patch or pixel, in terms of Model, Colour, Shape, Material and Texture.
The *Appearance Unit* may perform analysis using this description through a combination of other units' functionality.

### *D. Construct Axioms*

The following axioms are to represent constructs used within vision for modelling and as output to renderers and other modelling packages.

**Mesh**: This provides a centralised description of a mesh, with triangles, quads etc., subdivision, etc.
The *Mesh Unit* would provide IO mechanisms for meshes, construction from an image base (projective mesh) or from an arbitrary basis in 3D for reconstruction output.

**Grid**: This axiom gives descriptions of N-dimensional grids for use in vision. For example, a 2D grid would be a discretised image, a 3D grid would be a set of voxels, etc. The grid could be indexed at points (intersections of grid lines) or cells (volumes contained within grid edges).
The *Grid Unit* would provide methods of grid construction and conversion, along with IO mechanisms.

## IV. Algorithms and Problems

Based on the axioms previously defined, this section discusses how to use them to create researcher-level and developer-level abstractions. First we discuss the composition of sophisticated algorithms with the axioms using a bottom-up approach. Then we explore the idea of decomposing problems themselves through a top-down method into algorithms, rather than the reverse which is most often applied in the field. This leads to the idea of a formal description model which can be used to describe vision

problems at the highest level. We end with a discussion on the possibility of an interpreter which is capable of taking the description as input and translating that into the researcher-level algorithm space. This would allow developers without specialist vision knowledge to accomplish sophisticated tasks through knowledge of the problem they wish to solve.

### A. Algorithm Composition

The axioms of vision can be combined through a loose coupling (not direct inheritance, although that would also be possible) to form the basis of more sophisticated algorithms. The axioms themselves already encapsulate sophisticated concepts and algorithms, but by building on these we can create higher-level systems such as correspondence, 3D reconstruction, tracking etc.

For example, to create a simple 3D reconstruction system such as visual hull [18]: Matte may be used to create silhouette images; Camera provides calibration; Grid provides the volumetric construct; Transform gives the projected Grid points in the silhouettes to test for occupancy; Finally, Mesh creates the model for output from the volumetric grid.

A simple tracking could be also be accomplished with direct use of the axioms; however a more sophisticated version could add additional high-level components to build from (such as an accumulated state). The system could be initialised using Object, Matching and Similarity, and then through the use of Move and Optimise objects can be tracked through a sequence of images.

This level of use of the axioms represents the researcher-level abstraction, promoting code re-use and basic conceptual development. This could lead to a maintainable framework for the sharing of software.

### B. Problem Decomposition

The algorithm composition is not suitable for use by developers, as it requires expert knowledge of vision. Instead we should look at the top-most-level, which is the problem itself. If we can decompose a problem into smaller parts and use this as the basis for solving the problem, then we can open up computer vision to a much wider audience.

The idea behind this is to look at computer vision from a problem-centric viewpoint. If looking up a textbook to find out how to perform a 3D reconstruction with two narrow-baseline cameras, there should be a section which outlines this. Currently the algorithms are described and so a user with no particular knowledge of vision would not know where to look.

Again starting from the axioms, if the descriptions for each one could be formalised with a consistent model which relies only on knowledge of the concept and not the underlying algorithms or representations, then this would be accessible to a developer. Then higher-level components could also inherit these descriptions and add their own.

### C. Formal Description Model

Formally describing the problem is challenging in itself. For each axiom the problem it tries to solve must be defined in detail by considering the range of conditions under which the problem may appear. Using the Noise axiom as an example, detailing different types of noise (salt and pepper, Gaussian etc.), and associated properties such as amount, radius etc. is required.

At a higher level, consider the challenge of describing the problem of image registration. The problem is to spatially align a set of images to a common reference frame. The conditions under which this should be performed vary significantly: the images can be high-frequency or low-frequency; vary in focus, intensity and sensor type. Various algorithms can be used to solve the problem under these different conditions. If the user describes the data coming in under this problem space then an interpreter can take the description and translate it into an algorithm-centric description at the researcher-level. This would lead to an automatic vision system without the need for a vision specialist to implement it (beyond designing the interpreter and algorithms).

Additionally a user would need to describe the input and the expected output. Using registration again, if performing a panorama stitching, there is a reasonable expectation that the resulting transform will have a significant x-translation component, but probably little rotation, scaling and y-translation. The expected x-translation could be modelled as a distribution around the ideal expected solution with room for error around the ideal.

## V. Conclusions

Viewing the challenges of vision from the problem space will allow us to design systems which open up access to computer vision techniques to a much wider audience. In this paper we have presented the *axioms of vision*, a decomposition of the computer vision problem as a whole into small components which encapsulate core concepts. These axioms provide descriptions of their concept, and the implementations (units) provide functionality associated with the descriptions. Based on the axioms we can construct more sophisticated algorithms, maintain flexibility for research into new algorithms and promote re-use of existing code.

We also presented novel paradigms of thought for the computer vision problem. If we think in terms of the problem itself and its associated conditions, we can highlight new areas of research which haven't been explored as well as providing a coherent model which can be used by those not specialised in vision. The formal description model could extend to any problem in vision. We have presented the idea of an interpreter which would take the formal description and translate it such that the correct algorithm for the problem may be chosen.

We are currently working on all areas presented in this paper: we are implementing the units with the axiom descriptions to form the basis of an open source computer vision library; we are developing formal description models for many computer vision problems such as registration, tracking, correspondence and decomposition; we are also researching methods for automatic problem description extraction from images.

## REFERENCES


[1] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, 1st ed. O'Reilly Media, Inc., October 2008.

[2] Gandalf, "http://gandalf-library.sourceforge.net/."

[3] VXL, "http://vxl.sourceforge.net/."

[4] Camellia, "http://camellia.sourceforge.net/."

[5] A. Makarenko, A. Brooks, , and T. Kaupp, "On the benefits of making robotic software frameworks thin," in *International Conference on Intelligent Robots and Systems*, 2007.

[6] A. Afrah, G. Miller, D. Parks, M. Finke, and S. Fels, "Hive: A distributed system for vision processing," in *Proc. of the Int. Conf. on Distributed Smart Cameras*, September 2008.

[7] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, August 2002.

[8] Z. H., F. J.E., and G. S.A., "Image segmentation evaluation: A survey of unsupervised methods," *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 260–280, 2008.

[9] B. Zitov and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, pp. 977–1000, 2003.

[10] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, pp. 7–42, 2004.

[11] R. J. Campbell and P. J. Flynn, "A survey of free-form object representation and recognition techniques," *Computer Vision and Image Understanding*, vol. 81, no. 2, pp. 166–211, 2001.

[12] T. Yoo, M. J. Ackerman, W. E. Lorensen, W. Schroeder, V. Chalana, S. Aylward, D. Metaxes, and R. Whitaker, "Engineering and Algorithm Design for an Image Processing API," in *Medicine Meets Virtual Reality*, J. Westwood, Ed., 2002, pp. 586–592.

[13] M. Abramoff, P. Magelhaes, and S. Ram, "Image processing with ImageJ," *Biophotonics International*, vol. 11, pp. 36–42, 2004.

[14] J. Flusser, B. Zitov, and T. Suk, "Invariant-based registration of rotated and blurred images," in *in IEEE 1999 International Geoscience and Remote Sensing Symposium. Proceedings*. IEEE Computer Society, 1999, pp. 1262–1264.

[15] B. Zitov, J. Kautsky, G. Peters, and J. Flusser, "Robust detection of significant points in multiframe images," *Pattern Recogn. Lett.*, vol. 20, no. 2, pp. 199–206, 1999.

[16] C. Zitnick, S. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," in *Proc. of the Conf. on Computer Graphics and Interactive Techniques*, 2004, pp. 600—608.

[17] F. Mokhtarian and A. K. Mackworth, "A theory of multiscale, curvature-based shape representation for planar curves," *IEEE Transaction On Pattern Analysis and Machine Intelligence*, vol. 14, no. 8, pp. 789–805, 1992.

[18] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 2, pp. 150–162, 1994.