The Impact of Software Engineering on Image Processing

David B. Lowe, John R. Leaney
School of Electrical Engineering, University of Technology, Sydney
Ph: +61 2 330 2526, Fax: +61 2 330 2435
E-mail: {dbl, jleaney}@ee.uts.edu.au

Abstract

Image processing, in general, lacks the degree of formalism that is typically associated with more mature fields of research. The field is both relatively new and expanding rapidly. These two factors have led to developments outpacing the proper formalisation of existing knowledge and techniques. A major problem which has arisen is the lack of a proper understanding of the relationship between image information and image data. A number of efforts have been made to improve this situation (such as Marr's classical definition) but these tend to be either piecemeal, incomplete or outdated. This paper introduces aspects of systems/software engineering which have a bearing upon refining existing formalisms. The classical definition of image processing and the concepts of computational theory, algorithm and implementation are reformulated. It is argued that the application of software engineering to all phases of image processing will lead to reliable, predictable image processing systems. This will occur in three main areas; the specification and writing of software, the definition of image processing, and the development of a taxonomy of techniques and performance. This paper discusses these applications and presents preliminary work in this area.

1. Introduction

As ideas mature into disciplines so the playing in the sand pit has to mature to the building of structurally sound castles. It appears that now is an appropriate time for this to occur in image processing. Marr [1] is generally acknowledged as having made the first major steps in the area of general systematisation. He described three main principles. Firstly, the task must be specified in an unambiguous way. Secondly, an algorithm must exist to achieve this specification. And thirdly, their must be a plausible model which explains how a living visual apparatus achieves this process. Marr also defined a number of levels of vision. There are however a number of limitations to Marr's paradigm. Firstly, the work is still quite incomplete, and has not yet been taken far beyond Marr's original work. Secondly, Marr was considering a field much more narrow than is usually assumed to be the case; that of computer vision related to real-world scenes, as opposed to general image processing or analysis. These issues are yet to be addressed successfully.

Image processing suffers from a lack of structure. The most glaring consequence of this is that there exists a considerable discrepancy between the understanding of data transformations (usually well understood) and the understanding of the corresponding information transformation (often very poorly understood). Image processing depends heavily on specifying a systems requirements and then producing a correct, reliable, maintainable solution
for these requirements – the same as for software engineering. We believe that software engineering can be used to gain considerable insights into the systematisation of image processing.

Software engineering believes itself to have developed and refined what are basically techniques of systems engineering. Software engineering is an enabling technology; like systems engineering it assists in the process rather than the result – something which is also true of image processing. The authors see that software engineering can make contributions in three areas. Firstly, the application of the systems aspects of software engineering to a precise definition of what image processing is all about (basically refining Marr’s ideas). Secondly, the further application of the systems aspects to establish a taxonomy of techniques (such as performance indices, philosophical/psychological effects, set of rules/guidelines for mapping tools to requirements, techniques etc). Thirdly, the application of the software engineering discipline to the specification and writing of software for image processing in order to improve confidence in the implementation and results.

This paper initially gives a brief description of the aspects of software engineering which are salient to image processing (both in general, and its relationship to computational theory). The three areas of relevance of software engineering are then covered, followed by the conclusions that can be drawn from these discussions. We stress that this is just the beginning of ongoing work.

2. Image Processing and Software Engineering

2.1 Problems in Image Processing

Image processing is predominantly seen as involving the transformation of image data from one form (for example, a canonical pixel map) to another form (for example, a compressed representation of the image, a series of energy measures, or the spatial location of an object in the image). Algorithms are generally developed for particular applications by patching together techniques from a given toolset. The methods that are adopted are often ad hoc and few formal methods are adopted. The syntax is understood, but not necessarily the semantics.

A number of significant problems can be identified with current development methods. At a low level a given image processing algorithm can be viewed as a set of semi-formal processes that are primarily functional in nature. By semi-formal it is suggested that physical effects on data passed through a given process obey some mathematical rules or facts. For example, a filter performs a transformation which can be fully predicted by the mathematical properties of the filter. However, a major problem associated with this model is that little can be said with reference to the validity or appropriateness of the overall process, as implementing the desired information transformation (as opposed to the data transformation). This often results from the sheer quantity of the data being transformed, and the corresponding complex relationship between the data and the information which this data contains.

Algorithms and processing methods have been developed that produce (reasonably) well-defined results for a given set of inputs. However, there appears to be no real evidence that the functionality of the whole process has been defined in a form that may guarantee ‘correct’ results (or at least bounded results. What has resulted is that the image processing engineer has a toolkit of useful algorithms and programs each serving a valid (or at least deterministic) purpose, but the time and place to apply them is based on either heuristic choice or merely personal preference and familiarity.

Without a real understanding of the characteristics of image representations and the process there can be no hope of automated systems that operate beyond the barest applica-
tion domains. For this reason it is important to identify the valid domains for the tools presently available and the forms of image which will wholly support this goal. To remedy the present situation a solid theoretical examination of the processes and resultant data is required.

The second related major problem is more subtle, but more fundamental. The field as a whole has tended to lack structure. The overall dimensions of image processing and the structure of the field are not well defined. This tends to lead to considerable misdirection, a difficulty in understanding the field, poor standardisation, and minimal re-use within the field. Marr [1], and later Rosenfeld [2] have made some attempts at the systematisation of image processing, but these are very restricted in their scope and are correspondingly still incomplete.

2.2 Software Engineering

Software engineering is a field which has followed an analogous path to image processing, but several decades ahead. In its initial stages it was considerably ad hoc and has progressively developed (and is continuing to develop) into a well-defined discipline. It can make considerable contributions to solving the above problems, through analogy with its own development, and also through direct application of many of its principles and methodologies.

Software engineering has been given various definitions, though invariably these derive from the two definitions for software and engineering: "...software engineering is concerned with software systems built by teams rather than individuals, uses engineering principles in the development of these systems and included both technical and non-technical aspects" [3]. Software engineering has its roots in systems engineering, though it has developed its own specific set of tools and techniques. It predominantly revolves around a number of development phases (which may be sequential, iterative, or some other sequence, depending upon the life cycle model used). These phases are: requirements analysis and specification, system design, detailed design, implementation, testing, and maintenance.

Requirements analysis and specification is the process whereby the key aspects and characteristics of a system (or problem domain) are identified and specified. Design partitions the requirements and establishes an overall system architecture. Implementation realises the design as software. A series of different methodologies can be used for this process, with various tools for each methodology. Structured analysis and design analyses a system in terms of its data transformations and uses tools such as Data Flow Diagrams (DFD) and Data Dictionaries (DD). The object oriented methodology treats the individual components of the system as objects that encapsulate the object state and the functions that affect the object.

A number of ancillary issues within software engineering are quite relevant and worth mentioning. A number of methodologies make use (often implicitly) of what Meyer refers to as a contract [4]. This places pre and post conditions on operations, making explicit the functions performed. Software engineering also makes formal use of testing methodologies, performing various levels of testing; unit, integration, system, acceptance.

2.3 Application of Software Engineering to Image Processing

As discussed above, Marr [1] has made some attempts at the formalisation of the field of computer vision. He identifies three dimensions: the computational theory, the representation of the computational theory as an algorithm, and the implementation of the algorithm.
Software engineering has something to offer each of these three dimensions. Each one will be discussed in the following sections. Prior to this it is worthwhile noting that although these are being treated separately, an effective solution to many of these problems will rely on all aspects being addressed together.

3. Computational Theory: Definition Formalisation

Marr’s description of the computational theory is: “What is the goal of the computation, why is it appropriate, and what is the logic of the strategy by which it can be carried out?” This can be applied to the field overall and to the individual tasks within the field. What is the goal of image processing? Why is image processing appropriate for this particular problem? What is the logic behind the strategies used in image processing? These questions all relate to the overall definition of image processing in the context of computational theory. This covers its purpose, scope, and methodologies. Although these issues do not require software engineering, it can provide methods to formalise the answers; the use of formal specification languages (such as Z [5]), and the concepts developed for domain analysis.

The initial step in a software engineering problem is typically to identify and understand the problem domain. A large number of methods can be used to achieve this, but they are invariably attempting to achieve the same end result. This is to identify the aspects of the real world which relate to the system under consideration, and further to remove those aspects which are not relevant to the particular problem under consideration; i.e. we wish to develop a suitable model.

A similar approach can be taken with image processing. In order to understand what image processing can be capable of achieving, and how this is to be achieved, we need to first understand the domain of image processing. This can only occur if we fully understand the dimensions of the systematisation.

One of the major problems discussed above was the lack of a general structure to the image processing field. This can be partially overcome by the adoption of a suitable model. Within a processing framework, software engineering gives us two major models. The first is the data flow (or data transformation) model. Data is treated as transient. The second model is object oriented (or transaction) based. In this model, data is regarded as persistent and processes are clustered around the data that deal with it directly. The concept of state in the data becomes important. A study of these models in the context of image processing reveals that on a high level the data flow model predominates but at lower levels the object model pre-dominate.

These models can be used to gain an overall perspective of the functionality of the whole field and to formally define the bounds of the field. These bounds can be specified using such methods as pre and post conditions, and formal specification languages (such as Z). Rosenfeld [2] attempts to develop the paradigm further, producing models for both 2D and 3D image analysis. These models can easily be depicted using a Data Flow Diagram (DFD), showing the processing steps involved [6]. Unfortunately, Rosenfeld’s paradigms are specific to particular algorithms (they make explicit use of 2 1/2-D images, a representation which many image analysis schemes do not consider necessary [7]), and are oriented to the process, rather than the field as a whole. They are therefore not appropriate as a general model of the field.

We need to develop a formalisation of the field of image processing which is more complete than Marr’s, and more general than Rosenfeld’s. If we use Marr’s paradigm as a starting point, then we need to identify what we are trying to achieve conceptually, algorithmically, and practically. Figure 1 illustrates a first pass at this process. It shows the relationships
Figure 1: Image processing structure. This illustrates both the process and the object aspects of image processing. The process, as derived from Marr, transforms a set of requirements into an operational system. The object structure shows the relationships between the various data structures in the system. These two structures are orthogonal.

between the data flow model and the object model. These two models can both be used to describe the elements of the field, but are orthogonal in their structure. The next stage in the formalisation process would be to identify the components, bounds, and interactions of each of the elements shown in this figure.

Table 1 illustrates an initial indication of the content of the formalisation of this model (note that this is neither complete nor optimal). The relationships between the various phases and the elements incorporated into each phase can be used to define the structure of the field.

4. Representation and Algorithm: A Taxonomy

Marr describes representation and algorithm as "How can this computation theory be implemented? In particular what is the representation for the input and output, and what is the algorithm for the transformation?" Software engineering can help here by assisting in providing a taxonomy to formalise both representation and algorithm.

Once the definition of image processing has been formalised (as discussed in the previous section) we can then establish a taxonomy of techniques within this framework. A classification of techniques and tools will tend to significantly improve the current situation of continually re-inventing the wheel, and having problems specifying the overall behaviour of a system by looking at its constituent parts. It is important to note that the purpose of the taxonomy is not to catalogue processing algorithms (though this may well be a subset of the taxonomy) but rather to structure the field as a whole.
<table>
<thead>
<tr>
<th>Description</th>
<th>Theory</th>
<th>Algorithm</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>All aspects of computational theory: data representations, information content, and transformations from one data/information set to another.</td>
<td>Methods for obtaining both the data and information transformations required by the computational theory.</td>
<td>Implementations of the specific methods in such a way that ensures the correct data and information transformations.</td>
<td></td>
</tr>
<tr>
<td>Bounds</td>
<td>Should only specify what and why, not how. The information (rather than the data) being manipulated should be made explicit.</td>
<td>Specifies how, not what. Should not make any interpretation about either data or information transformation.</td>
<td>Implements the explicitly specified algorithm and nothing more.</td>
</tr>
<tr>
<td>Data</td>
<td><strong>Input</strong>: Formal user requirements  <strong>Output</strong>: Formal data representation description and associated information and data transformation required.</td>
<td><strong>Output</strong>: A set of algorithms and data structures that satisfy the required transformations.</td>
<td><strong>Output</strong>: A system that satisfies the requirements</td>
</tr>
</tbody>
</table>

Table 1: Image processing structure description.

At this stage in this work we do not wish to construct this taxonomy, but rather identify what benefits it will bring and how it should be structured. This is an ongoing process within software engineering. New paradigms, models, methodologies and tools are continually being developed. The primary difference between this development and that in image processing is that the developments within software engineering generally (though admittedly not always) fit into the overall structure of the field in a well-defined manner, and have a quantifiable purpose, structure, and relationship to other developments within the field. Image processing needs a taxonomy which matches this situation.

### 4.1 Taxonomy Structure

Prior to developing a taxonomy for image processing we need to consider the requirements of such a taxonomy. As mentioned previously, the development of software engineering is analogous to that which image processing is now undergoing. We should be able to identify aspects of software engineering which can give us insights into image processing. Software engineering originally did not exist *per se*. Rather it has gradually developed from the recognition of problems with computer programming. These problems are similar to those currently being experienced with image processing. Software was initially developed very much on an ad hoc basis, much as image processing algorithms are developed now. The software (and the associated algorithms) gradually became more complex, and in the process the development and maintenance of this software became more important. Current image processing tools (e.g. imaging hardware) and methods (e.g. filters) correspond to the tools (e.g. compilers, linkers) and algorithms (e.g. sorting algorithms) used in software engineering. Yet this is only one aspect of software engineering.

The taxonomy of software engineering includes not only the area of implementation tools and techniques, but also the full life cycle of a software development. The same should apply to image processing. Using as an initial starting point, parallels between software engineering and image processing a typical skeletal taxonomy can be identified. This is not intended to be definitive nor complete, but simply an example.
• Theory: Human Visual System (physiology, psychology, biology), Applications (vision, remote sensing, transmission, storage, enhancement ...), Algorithms (filtering, feature extraction, quality calculation) etc.

• Process: Analysis (understanding of the domain, requirements), Design, Implementation, Testing, Maintenance, etc.

• Representation: Data formats, Information models, etc.

• Structure:

4.2 Taxonomy Benefits

By developing a comprehensive, well-structured, understandable taxonomy of image processing a number of significant benefits can be obtained. These are as follows:

Process validity and appropriateness: Although in most cases at present the data transformations are understood, the information transformations are not. The development of appropriate taxonomy will enable the information transformations to be understood considerably more readily. This will allow the computational validity of the process, and its appropriateness to the particular problem to be established.

Understandability: Related to the above comment on information transformation is that the processes and results obtained are much more readily understood.

Standardisation: A taxonomy will facilitate the standardisation of many aspects of the field, improving productivity both in development and research.

Modularisation and Re-use: Again, improving productivity, will be the increased ability to reuse existing solutions or parts of solutions. The current situation where many problems are solved (or attempted to be solved) by using inappropriate tools will be removed.

5. Implementation: Specification, Design and Writing of Software

Marr's description of this dimension is "How can this representation and algorithm be realised physically?" Software engineering provides tools that provide considerable assistance in the implementation of software (and hardware) systems. Most of the image processing algorithms that are developed are implemented in software (only simple algorithms such as filters, and transformations are implemented in hardware).

The development of this software can benefit immediately from software engineering. These benefits are detailed in length in almost any book on software engineering (see for example [3]). Briefly, software engineering can improve the software by:

• Reducing the development time, through the reduction of implementation, testing and maintenance time, through increased re-use, and improved understanding.

• Producing higher quality software. This includes software which is more correct, efficient, maintainable, portable, reliable, reusable, robust, and user-friendly.

• An increased confidence that the software system that is developed is solving the appropriate problem.

• An increased understanding of the process and risks associated with the system.
6. Conclusions

We have attempted to identify problems with the current state of image processing as a field, and then illustrate how software engineering can offer many insights into possible solutions to these problems. The problems relate mainly to the lack of an overall structure to the field. Problem definitions, algorithms intended to solve these problems, and systems implementing these algorithms are poorly structured and subsequently poorly understood. The development is often ad hoc. The most glaring of these problems can be described by considering the discrepancy between the understanding of data transformations (usually well understood) and the understanding of the corresponding information transformation (often very poorly understood). As a result there is often poor comprehension of the validity of appropriateness of a particular process.

Image processing offers insights into three aspects of this problem. Firstly it can help in the overall formalisation of the definition of the field and its scope. Secondly it can assist in the development of an appropriate taxonomy for the field, bringing considerable benefits in doing so. Finally, software engineering can be used in the specification and development of software for use in image processing.

This paper does not attempt to solve any of the problems raised. Rather it is trying to raise the awareness of the existence of these problems and point out methods that may be used to solve them. This work is only in its initial stages and will continue to develop along the lines outlined in the paper.

The first steps to be taken will be in formalising further the definition of the field of image processing. This will revolve around the formal identification and definition of the components, bounds, and interactions of each of the elements of the image processing model that is being developed. Once this has been achieved this model can be used in the development of a structured taxonomy for the field. An initial indication of how this might be achieved is given but, again, this needs to be formalised. In conjunction with each of these tasks will be the adoption of suitable software engineering methodologies in the development of image processing software.

References