# Semantics and Planning Based Workflow Composition for Video Processing[*]

Gayathri Nadarajan
Yun-Heh Chen-Burger Robert B. Fisher

Accepted: April 16th 2013

**Abstract**

This work proposes a novel workflow composition approach that hinges upon ontologies and planning as its core technologies within an integrated framework. Video processing problems provide a fitting domain for investigating the effectiveness of this integrated method as tackling such problems have not been fully explored by the workflow, planning and ontological communities despite their combined beneficial traits to confront this known hard problem. In addition, the pervasiveness of video data has proliferated the need for more automated assistance for image processing-naive users, but no adequate support has been provided as of yet.

The integrated approach was evaluated on a video set originating from open sea environment of varying quality. Experiments to evaluate the efficiency, adaptability to user's changing needs and user learnability of this approach were conducted on users who did not possess image processing expertise. The findings indicate that using this integrated workflow composition and execution method: 1) provides a speed up of over 90% in execution time for video classification tasks using full automatic processing compared to manual methods without loss of accuracy; 2) is more flexible and adaptable in response to changes in user requests than modifying existing image processing programs when the domain descriptions are altered; 3) assists the user in selecting optimal solutions by providing recommended descriptions.

[*] This research was funded by European Commission FP7 grant 257024, in the Fish4Knowledge project (www.fish4knowledge.eu).

# 1 Introduction

Traditional workflow systems have several drawbacks, *e.g.* in their inabilities to rapidly react to changes, to construct workflow automatically (or with user involvement) and to improve performance autonomously (or with user involvement) in an incremental manner according to specified goals. Overcoming these limitations would be highly beneficial for complex domains where such adversities are exhibited. Video processing is one such domain that increasingly requires attention as larger amounts of images and videos are becoming available to those who are not technically adept in modelling the processes that are involved in constructing complex video processing workflows. The requirements to address the problem of automated video analyses for non-expert users include i) process automation; ii) rich process modelling; iii) performance-based tool selection; and iv) adaptable to changing user needs.

Conventional video and image processing (VIP) systems, on the other hand, are developed by image processing experts and are tailored to produce highly specialised hand-crafted solutions for very specific tasks, making them rigid and non-modular. Traditionally they produce single-executable systems that work accurately on a specific set of data. The knowledge-based vision community have attempted to produce more modular solutions by incorporating ontologies. However, they have not been maximally utilised to encompass aspects such as application context descriptions (*e.g.* lighting and clearness effects) and qualitative measures.

This work aims to tackle some of the research gaps yet to be addressed by the workflow and knowledge-based image processing communities by proposing a novel workflow composition approach within an integrated framework. This framework distinguishes three levels of abstraction via the design, workflow and processing layers. The core technologies that drive the workflow composition mechanism are ontologies and planning. Video processing problems provide a fitting domain for investigating the effectiveness of this integrated method as tackling such problems have not been fully explored by the workflow, planning and ontological communities despite their combined beneficial traits to confront this known hard problem. In addition, the pervasiveness of video data has proliferated the need for more automated assistance for image processing-naive users, but no adequate support has been provided as of yet.

A set of modular ontologies was constructed to capture the goals, video descriptions and capabilities (video processing tools). They are used in conjunction with a domain independent planner to help with performance-based selection of solution steps based on preconditions, effects and postconditions.

Two key innovations of the planner are the ability to support workflow execution (by interleaving planning with execution) and can perform in automatic or semi-automatic (interactive) mode. In the interactive mode, the user is involved in tool selection based on the recommended descriptions provided by the workflow system via the ontology. Once planning is complete, the result of applying the tool of their choice is presented to the user visually for verification. This plays a pivotal role in providing the user with control and the ability to make informed decisions. Video processing problems can also be solved in more modular, reusable and adaptable ways compared to conventional image processing systems.

The integrated approach was evaluated on a test set consisting of videos originating from open sea environment of varying quality. Experiments to evaluate the efficiency, adaptability to user's changing needs and user learnability of this approach were conducted on users who did not possess image processing expertise. The findings indicate that using this integrated workflow composition and execution method: 1) provides a speed up of over 90% in execution time for video classification tasks using full automatic processing compared to manual methods without loss of accuracy; 2) is more flexible and adaptable in response to changes in user requests (be it in the task, constraints to the task or descriptions of the video) than modifying existing image processing programs when the domain descriptions are altered; 3) assists the user in selecting optimal solutions by providing recommended descriptions.

**Outline.** Section 2 describes existing workflow composition initiatives and highlights their shortfalls. The workflow composition framework is outlined in Section 3, which introduces the workflow tool and its components. One of the major component, the ontology will be detailed in Section 4. The second core technology, the planning mechanism is explained in Section 5. Evaluation of the overall approach and analysis are provided in Section 6. Future directions for this research is concluded in Section 7.

## 2 Grid Workflow Systems

With the advent of distributed computing in the past two decades, workflows have been deployed in distributed platforms. In a distributed context, such as the Grid or e-Science [9] a workflow can be abstracted as a composite web service, *i.e.* a service that is made up of other services that are orchestrated in order to perform some higher level functionality. The goal of e-Science workflow systems is to provide a specialised programming environment to simplify the programming effort required by scientists to orchestrate a computational science experiment [31]. Therefore, Grid-enabled systems must facilitate the composition of multiple resources, and provide mechanisms for creating and enacting these resources in a distributed manner. This requires means for **composing** complex workflows for execution, which has attracted considerable effort within the Grid workflow community.

Several major workflow systems were analysed in terms of workflow composition. Among them include Pegasus [7], a workflow management system that aims to support large-scale data management in a variety of applications such as astronomy, neuroscience, biology, gravitational wave-science and high-energy physics, Triana [32, 33], a problem solving and workflow programming environment that has been used for text, speech and image processing tasks, Taverna [27], an open source workflow engine that aims to provide a language and software tools to facilitate easy use of workflow and distributed compute technology for biologists and bioinformaticians and Kepler [17], a workflow project that consists of a set of Java packages supporting heterogeneous, concurrent modelling to design and execute scientific workflows. Workflow composition mechanisms and limitations of current efforts will be discussed in the following subsections.

## 2.1 Workflow Composition Mechanisms

Studies on workflow systems have revealed four aspects of the workflow lifecycle – composition, mapping (onto resources), execution and provenance capture [6]. This paper focuses on the composition and execution aspects of the workflow lifecycle. Workflow composition can be textual, graphical or semantics-based. Textual workflow editing requires the user to describe the workflow in a particular workflow language such as BPEL [1], SCUFL [26], DAGMan [29] and DAX [7]. This method can be extremely difficult or error-prone even for users who are technically adept with the workflow language. Graphical renderings of workflows such as those utilised by Triana, Kepler and VisTrails [3] are easy for small sized workflows with fewer than a few dozen tasks. However many e-Science and video processing workflows are more complex. Some workflows have both textual and graphical composition abilities. The CoG Kit's Karajan [35] uses either a scripting language, GridAnt or a simple graphical editor to create workflows.

Blythe *et al.* [2] have researched into a planning-based approach to workflow construction and of declarative representations of data shared between several components in the Grid. This approach is extendable to be used in a web services context. Workflows are generated semi-automatically with the integration of the Chimera system [10]. In Pegasus [7], abstract workflows[1] may be constructed with the assistance from a workflow editor, such as the Composition Analysis Tool (CAT) [16] which critiques partial workflows composed by users and offers suggestions to fix composition errors and to complete the workflow templates. It assumes that the user may not have the explicit descriptions of the desired goals at the beginning. It utilises classical planning to perform workflow verification. Wings [12] extends this by dealing with the creation and validation of very large scientific workflows. However, CAT requires the user to construct a workflow before interactively verifying it to produce a final workflow. Our effort, in contrast, aims to construct the workflow interactively or automatically.

Splunter *et al.* [34] propose a fully automated agent-based mechanism for web service composition and execution using an open matching architecture. In a similar vein to these two approaches, our effort aims to provide semi-automatic and automatic means for workflow composition, but does not deal with the mapping of resources onto the workflow components.

Three distinct stages are distinguished for workflow creation; the creation of workflow templates, the creation of workflow instances and the creation of executable workflows (done by Pegasus). Workflow templates specify complex analyses sequences while workflow sequences specify data. Workflow templates and instances are semantic objects that are represented in ontologies using OWL-DL. While semantics-based workflow composition is the subject of current research, most efforts have focused on either easing the task of large scale workflows creation for computational workflows and for web services [6].

## 2.2 Limitations of Current Grid Workflow Solutions

The major workflow systems mentioned at the beginning of the section possess some features that are worth investigating in order to assess their suitability and limitations for the purposes of this study. In terms of composition itself, Pegasus's main strength is in mapping abstract workflows to their concrete (executable) forms, which are then executed by a scheduler. It also provides adaptivity through a partitioner that uses planning to produce partial executable workflows. It does not, however, provide automatic composition of the abstract workflows.

Triana, Taverna and Kepler contain similar elements; Triana's tasks are conceptually the same as Taverna's processes and Kepler's actors. The approach in Kepler is very similar to Triana in that the workflow is visually con-

[1] Directed acyclic graphs (DAGs) composed of tasks and data dependencies between them.

structed from actors (Java components), which can either be local processes or can invoke remote services such as Web services. In terms of applicability, Pegasus would best suit a domain with well-defined requirements and where the overall goal could be determined from a given set of rules and constraints. Triana is well-suited for composing complex workflows for Web services and Peer to Peer services. Taverna is also suitable to be used in Web and Grid services contexts, but its use may be limited to composing simple workflows, whereas Kepler works very well for composing workflows for complex tasks but it has yet to reach its potential as a fully Grid-enhanced system. Kepler is built upon Ptolemy II which is primarily aimed at modelling concurrent systems. Furthermore, it is designed to be used by scientists which imposes some level of expertise to the user.

While existing workflow systems have more recently incorporated ontologies, their use is still limited. The use of such technologies should not be exclusively independent, rather they should be fully integrated into the system. Existing systems do not provide full ontological handling nor integration, instead they make use of separate ontology tools to define and manipulate ontologies. The main limitations of existing workflow initiatives can be summarised as follows:

- Limited or no automated support in constructing workflows, thus requiring the user to possess domain expertise.
- Unable to improve performance autonomously (or with user involvement) in an incremental manner according to specified goals.
- Generally do not have full integration of ontologies that would allow for more powerful representation and reasoning abilities.

Next, the framework constructed to overcome the limitations of existing efforts, in particular to provide automatic workflow composition is described.

## 3 Hybrid Three-layered Workflow Composition Framework

A hybrid semantics-based workflow composition method within a three-layered framework was devised and implemented (Fig. 1). It distinguishes three different levels of abstraction through the design, workflow and processing layers. Each layer contains several key components that interact with one another and with components in other layers. This has been the backbone of the (**S**emantics-based **W**orkflows for **A**nalysing **V**ideos) (SWAV) tool [21].

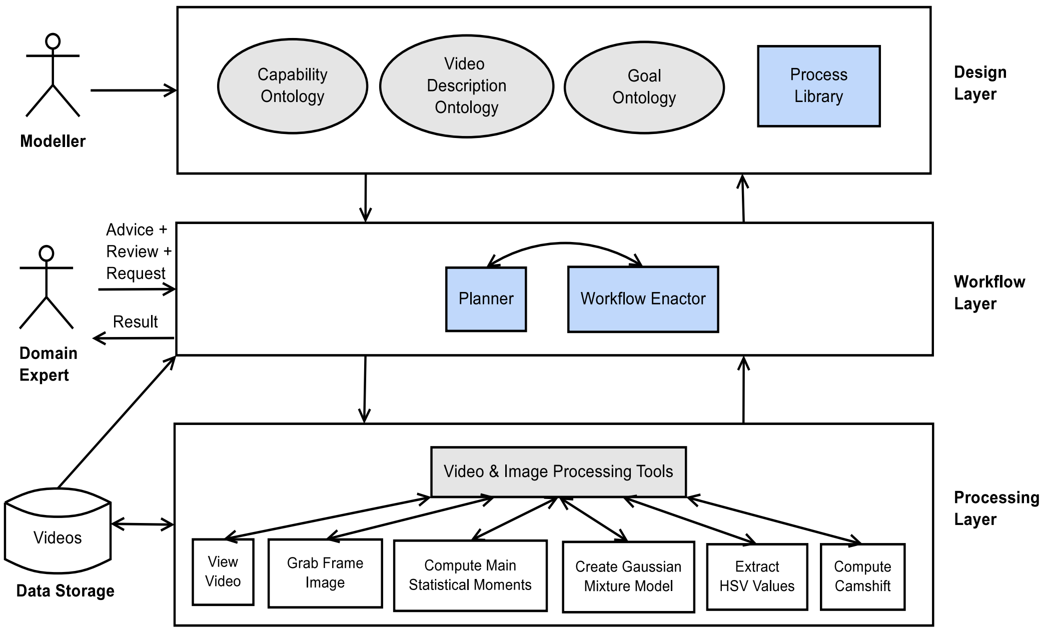


Figure 1: Overview of hybrid workflow composition framework for video processing. It provides three levels of abstraction through the design, workflow and processing layers. The core technologies include ontologies and a planner, used in the SWAV tool.

The **design layer** contains components that describe the domain knowledge and available video processing tools. These are represented using ontologies and a process library. A modeller is someone who is able to manipulate the components of the design layer, for example populate the process library and modify the ontologies. Typically the modeller has training in conceptual modelling and has knowledge in the application domain, but not necessarily. The components could also be updated automatically, as will be shown in Section 5. Knowledge about image processing tools, user-defined goals and domain description is organised qualitatively and defined declaratively in this layer, allowing for versatility, rich representation and semantic interpretation. The ontologies which are used for this purpose will be described in Section 4.

The process library developed in the design layer of the workflow framework contains the code for the image processing tools and methods available to the system. These are known as the process models. The first attempt in populating the process library involved identifying all primitive tasks for the planner based on the finest level of granularity. A primitive task is one that is not further decomposable and may be performed directly by one or more image processing tools, for instance a function call to a module within an image processing library, an arithmetic, logical or assignment operation. Each primitive task may take in one or more input values and return one or more output values. Additionally, the process library contains the decomposition of non primitive tasks or *methods*. This will be explained in Section 5. The complete list of independent executables can be found in [20].

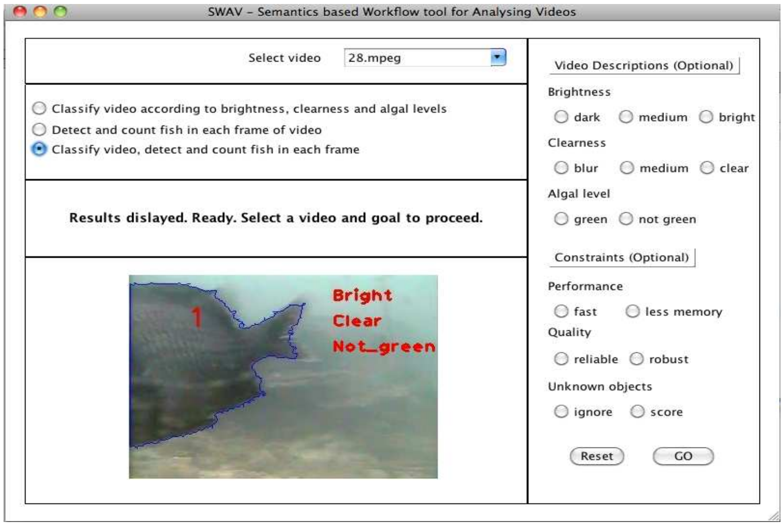
The **workflow layer** is the main interface between the user and the system. It also acts as an intermediary between the design and processing layers. It ensures the smooth interaction between the components, access to and from various resources such as raw data, image and video processing toolset, and communication with the user. Its main reasoning component is an execution-enhanced planner that is responsible for transforming the high level user requests into low level video processing solutions. Detailed workings of the planner is contained in Section 5.

The workflow enactor plays the important role of choreographing the flow of processing within the system. It should be noted that unlike the workflow enactors covered in Section 2, the SWAV tool does not deal with resource allocation and scheduling, rather, on the composition of specific operators and the execution of the operators given their predetermined parameters. First it reads in the user request in textual form (use selects from a list of options). Next it consults the goal and video description ontologies to formulate the input that is then fed to the planner. When the planner, with the assistance of the process library and capability ontology, returns the final solution plan, the enactor prompts the user for further action. The user has access to the final result of the video processing task visually, and has the choice to 1) rerun the same task on the same video but with modifications to the domain information; 2) rate the quality of the result; or 3) perform another task. The composed workflow is saved in a script file that can be invoked off-line. By being able to view the result of each solution with changes to the domain information, the user can assess the quality of the solution produced. This feedback mechanism could be used as a basis for improving the overall performance of the system as verifying the quality of the video processing solutions automatically is not a trivial task. The planning mechanism is described in Section 5.

The **processing layer** consists of a set of video and image processing tools that can perform various image processing functions. The functions of these tools are represented in the capability ontology in the design layer. Once a tool has been selected by the planner, it is applied to the video directly. The final result is passed back to the workflow layer for output and evaluation.

The functions (primitive tasks) that they can perform are represented semantically in the capability ontology, described in Section 4. A set of video processing tools developed for this research is available at [20].

Several prototypes of the workflow interface were designed over time; initially it uses a textual interface to communicate with the user and more recently the SWAV tool incorporates a graphical interface (Fig. 2).



SWAV – Semantics based Workflow tool for Analysing Videos

Select video 28.mpeg

○ Classify video according to brightness, clearness and algal levels
○ Detect and count fish in each frame of video
● Classify video, detect and count fish in each frame

**Results dislayed. Ready. Select a video and goal to proceed.**

Video Descriptions (Optional)

Brightness
○ dark ○ medium ○ bright

Clearness
○ blur ○ medium ○ clear

Algal level
○ green ○ not green

Constraints (Optional)

Performance
○ fast ○ less memory

Quality
○ reliable ○ robust

Unknown objects
○ ignore ○ score

Figure 2: The SWAV interface which allows user to select a video and a task (top left panels) and adjust domain settings (right panel). Annotated results are displayed to the user.

# 4 Video and Image Processing Ontology

Ontologies are used for capturing knowledge and semantics in a domain and have been used widely in several major fields including medical, linguistics and enterprise. Domain ontologies are often modelled in a collaborative effort between domain and ontology experts to capture *consensual* knowledge that is formed between the domain experts that can be shared and reused among them. In the video processing field, ontologies are extremely suitable to many problems that require prior knowledge to be modelled and utilised in both a descriptive and prescriptive capacity since they encode the concepts and relationships between the components in the world.

## 4.1 Modularisation

For the purposes of this research, a set of ontologies was required to model the video and image processing (VIP) field so that it can be used for domain description and understanding, as well as inference. The ontology should describe the domain knowledge and support reasoning tasks, while being reasonably independent from the system. The principles adopted for the ontology construction included simplicity, conciseness and appropriate categorisation. For this reason, three aspects of the VIP field were highlighted. These were identified as *goal, video description* and *capability.* These aspects were motivated by the context of their use within a planning system that requires the goal and initial domain state model (which includes the initial video description) and also a performance-based selection of operators. Following the SUMO (Suggested Upper Merged Ontology)[2] ontology representation, a modular ontology construction was adopted (see Fig. 3). The modularisation aims to separate the formulation of the problems from the description of the data and the solutions to be produced.

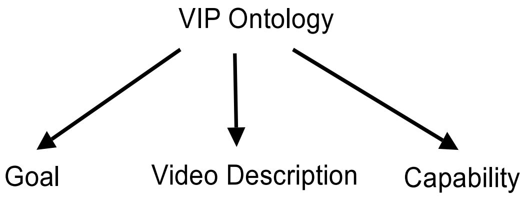


Figure 3: Modular structure of the Video and Image Processing (VIP) Ontology.

The next three subsections describe the three ontologies, a more detailed explanation of their construction in relation to the Fish4Knowledge project can be found in [19].

## 4.2 Goal Ontology

The Goal Ontology contains the high level questions posed by the user and interpreted by the workflow as VIP tasks, termed as goals, and the constraints to the goals. The main concepts of the goal ontology is shown in Fig. 4.

Under the 'Goal' class, more specialised subclasses of video processing goals are described. Some examples include 'Object detection', 'Event detection' and 'Object Clustering'. Under each of these, even more specialised goals are described. For instance, more specific goals of 'Object detection'

[2] http://www.ontologyportal.org/

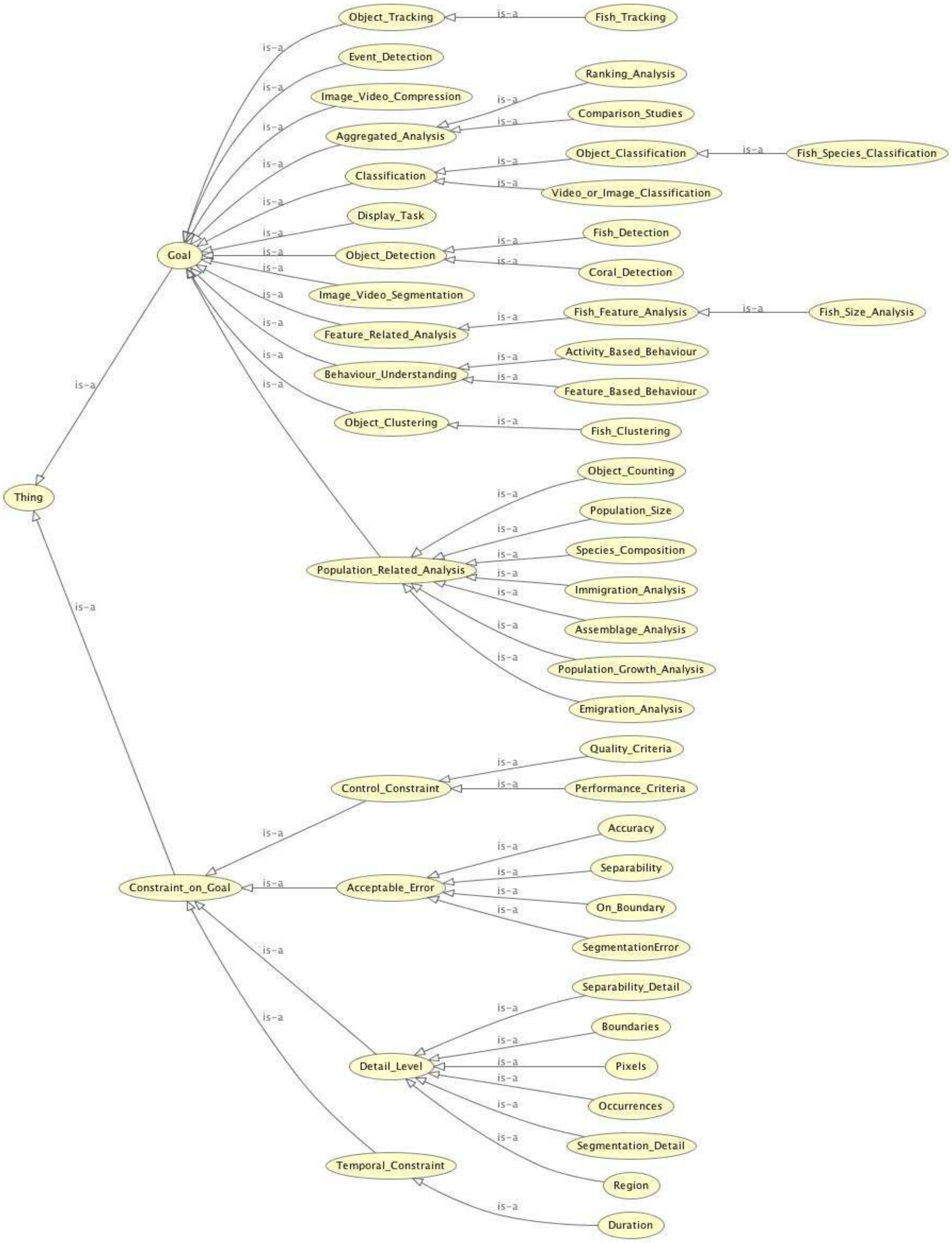


Figure 4: Goal ontology denoting the main classes of goals and constraints.

include 'Fish detection' and 'Coral detection', which are relevant for this work.

‘Constraint on Goal’ refers to the conditions that restrict the video and image processing tasks or goals further. In our context, the main constriction for a VIP goal is the ‘Duration’, a subclass of ‘Temporal Constraint’. Each task may be performed on all the historical videos, or a portion specified by the user – within a day, night, week, month, year, season, sunrise or sunset (all specified as instances of the class ‘Duration’).

Other constraints types include ‘Control Constraint’, ‘Acceptable Error’ and ‘Detail Level’. The control constraints are those related to the speed of VIP processing and the quality of the results expected by the user. ‘Performance Criteria’ allows the user to state whether the goal that they wish to perform should be executed using a faster algorithm (indicated by the criterion processing time) or whether it should take less (CPU) memory. Processing time and memory are instances of ‘Performance Criteria’. Instances of the class ‘Quality Criteria’ are reliability and robustness. ‘Quality Criteria’ with the value reliability constrains the solution to be the most accurate result. If such a solution could not be found, then the system should fail rather than produce alternative options. ‘Robustness’ indicates the reverse; that the system should not break down completely in cases where a reliable solution could not be found, instead it should return an alternative (imperfect) result.

Another constraint that the user may want to insert is the threshold for errors, contained in the class ‘Acceptable Error’. A typical example of this is contained in its subclass ‘Accuracy’, which states the accuracy level of a detected object. Two instances of accuracy are prefer miss than false alarm’ and prefer false alarm than miss. Miss and false alarm are terminologies used within VIP tasks that involve the detection of objects to indicate the accuracy level of the detection. Consider a real object to be the object that needs to be detected. A miss (false negative) occurs when a real object exists but is not detected. A false alarm (false positive) occurs when an object that is not a real object has been detected.

The class ‘Detail Level’ contains constraints that are specific to particular details, for example detail of ‘Occurrence’. The criteria for ‘Occurrence’ is used for detection tasks to constrict the number of objects to be detected. The value ‘all’ for occurrences imposes that all the objects should be identified.

The Goal Ontology is used for consistency checks when a user query is detected in the system. It can check that the query matches with a goal or set of goals that is achievable within the workflow system. It is also used to guide the selection of higher level tasks for workflow and formulate input values to the reasoning engine that is responsible for searching the VIP solution set for a VIP task, *i.e.* to compose the workflow.

## 4.3 Video Description Ontology

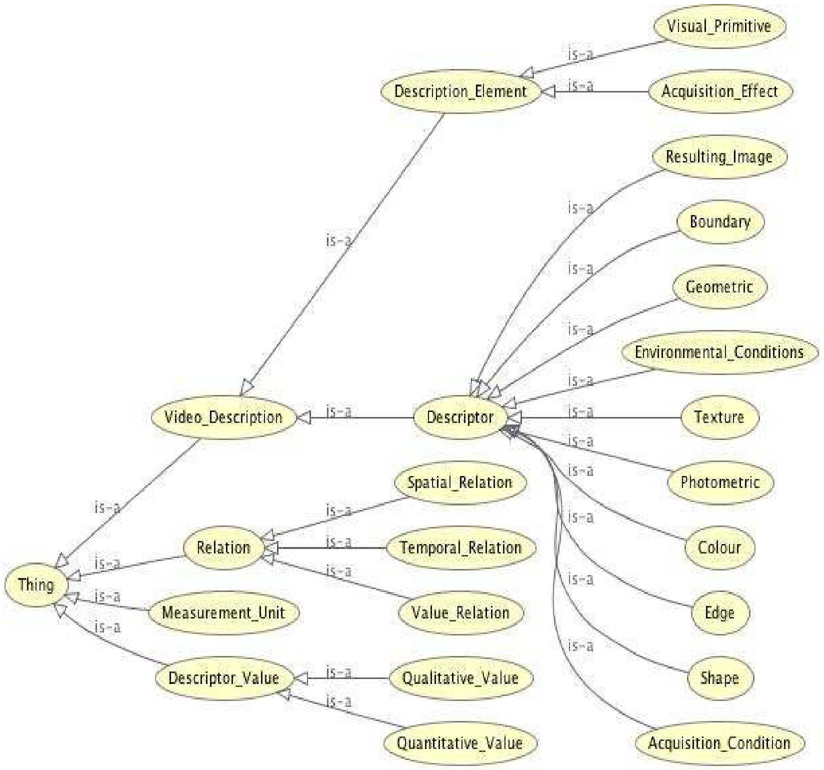


Figure 5: Main concepts of the Video Description Ontology.

The Video Description Ontology describes the concepts and relationships of the video and image data, such as what constitutes video/image data, the acquisition conditions such as lighting conditions, colour information, texture, **environmental conditions** as well as spatial relations and the range and type of their values. Fig. 5 gives a pictorial overview of the main components of the Video Description Ontology. The upper level classes include ‘Video Description’, ‘Descriptor Value’, ‘Relation’, and ‘Measurement Unit’.

The main class of this ontology is the ‘Video Description’ class, which has two subclasses – ‘Description Element’ and ‘Descriptor’. A description element can be either a ‘Visual Primitive’ or an ‘Acquisition Effect’. A visual primitive describes visual effects of a video/image such as observed object’s geometric and shape features, *e.g.* size, position and orientation while acquisition effect descriptor contains the non-visual effects of the whole video/image that contains the video/image class such as the brightness (luminosity), hue and noise conditions. The descriptor for the description elements are contained under the ‘Descriptor’ class and are connected to the

'Description Element' class via the object property 'hasDescriptionElement' (not visible in the diagram).

Typical descriptors include shape, edge, colour, texture and environmental conditions. Environmental conditions, which are acquisitional effects, include factors such as current velocity, pollution level, water salinity, surge or wave, water turbidity, water temperature and typhoon, specified as instances. These values that the descriptors can hold are specified in the 'Descriptor Value' class and connected by the object property 'hasValue'. For the most part, qualitative values such as 'low', 'medium' and 'high' are preferred to quantitative ones (*e.g.* numerical values). 'Qualitative' values could be transformed to quantitative values using the 'convertTo' relation. This would require the specific measurement unit derived from one of the classes under the concept 'Measurement Unit' and conversion function for the respective descriptor *e.g.* a low velocity could be interpreted as movement with velocity within a range of 0 and $25ms^{-}1$[3]. Some descriptor values can be tied to their appropriate measurement units. The property that specifies this is 'hasMeasurementUnit', which relates instances in the class 'Descriptor' to instances in the class 'Measurement Unit'.

The goal and video description ontology were developed with collaboration with image processing experts, knowledge-based vision communities and domain experts. Preliminary work with the Hermes project [28] brought about initial versions of these two ontologies [22].

## 4.4 Capability Ontology

The capability ontology (Fig. 6) contains the classes of video and image processing functions and tools. Each function (or capability) is associated with one or more tools. A tool is a software component that can perform a video or image processing task independently, or a technique within an integrated vision library that may be invoked with given parameters. This ontology will be used directly by the planner in order to identify the tools that will be used to solve the problem. As this ontology was constructed from scratch, the METHONTOLOGY methodology [13] was adopted. It is a comprehensive methodology for building ontologies either from scratch, reusing other ontologies as they are, or by a process of re-engineering them. The framework enables the construction of ontologies at the knowledge level, *i.e.* the conceptual level, as opposed to the implementation level.

[3] Currently, there is not a fixed conversation formula. The actual conversion formula used will be determined as we gain more experience by using our workflow system over time.

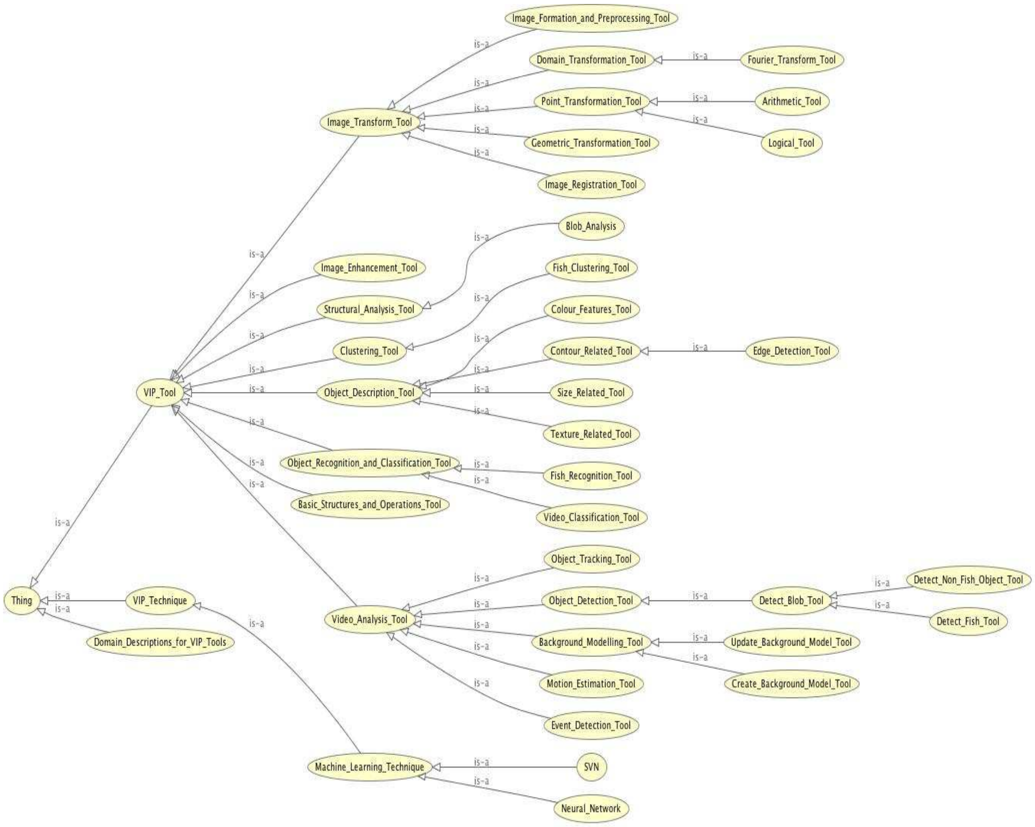


Figure 6: Main concepts of the Capability Ontology.

This ontology will be used to identify the tools that will be used for workflow composition and execution of VIP tasks. In our context, it is used by a reasoner and a process library for the selection of optimal VIP tools. The main concepts intended for this ontology have been identified as ‘VIP Tool’, ‘VIP Technique’ and ‘Domain Descriptions for VIP Tools’. Each VIP technique can be used in association with one or more VIP tools. ‘Domain Description for VIP Tool’ represent a combination of known domain descriptions (video descriptions and/or constraints to goals) that are recommended for a subset of the tools. This will be used to indicate the suitability of a VIP tool when a given set of domain conditions hold at a certain point of execution. At present these domain descriptions are represented as strings and tied to VIP tools, e.g. Gaussian background model would have the description ‘Clear and Fast Background Movement’ to indicate the best selection criteria for it.

The main types of VIP tools are video analysis tools, image enhancement tools, clustering tools, image transform tools, basic structures and operations tools, object description tools, structural analysis tools and object recognition and classification tools. At present fish detection and tracking have been performed more than the other tasks, hence the ontology has been populated with most of the tools associated with these tasks. For other tasks that have not been performed, *e.g.* fish clustering, the ontology will be extended and populated in due course. Detection and tracking tools fall under the class 'Video Analysis Tool'. Other types of video analysis tools are event detection tools, background modelling tools and motion estimation tools.

The class 'Object Description Tool' specifies tools that extract features such as colour, texture, size and contour, while image transform tools are those concerned with operations such as point, geometric and domain transformations. 'VIP Technique' is a class that contains technologies that can perform VIP operations. For now, two types of machine learning techniques have been identified. These techniques could be used to accomplish the task of one or more VIP tools. For example, neural netwworks can be used as classifiers as well as detectors.

The Capability Ontology can be used for reasoning during workflow composition using planning. As planning takes into account preconditions before selecting a step or tool, it will assess the domain conditions that hold to be used in conjunction with an appropriate VIP tool.

## 4.5 Walkthrough

Based on the devised sub-ontologies, a walkthrough of their usage to provide different levels of vocabulary in a seamless and related manner is explained here. The flowchart in Fig. 7 outlines the workflow enactment process emphasising on points that require ontological usage.

The user may have a high level goal or task such as *"Detect all the fish in the video 1.mpeg"* in mind. One way this could be represented and selected are via the following criterion-value pairs in natural language:

[Goal: goal = fish_detection]
[Constraints: Performance = memory, Quality = reliability,
Accuracy = prefer_miss_than_false_alarm, Occurrence = all]
[Video Descriptions: Brightness = not_known, Clearness = not_known,
Green Tone = not_known]

As a first step, the user selects the goal s/he wishes to conduct via the workflow interface. This corresponds to steps 1 and 2 in the flowchart. Once the goal is retrieved, it is checked against the goal ontology (step 4). Axioms in the goal ontology are used to check if the goal is indeed one that is valid for the application in question. In this case the instance 'fish_detection'

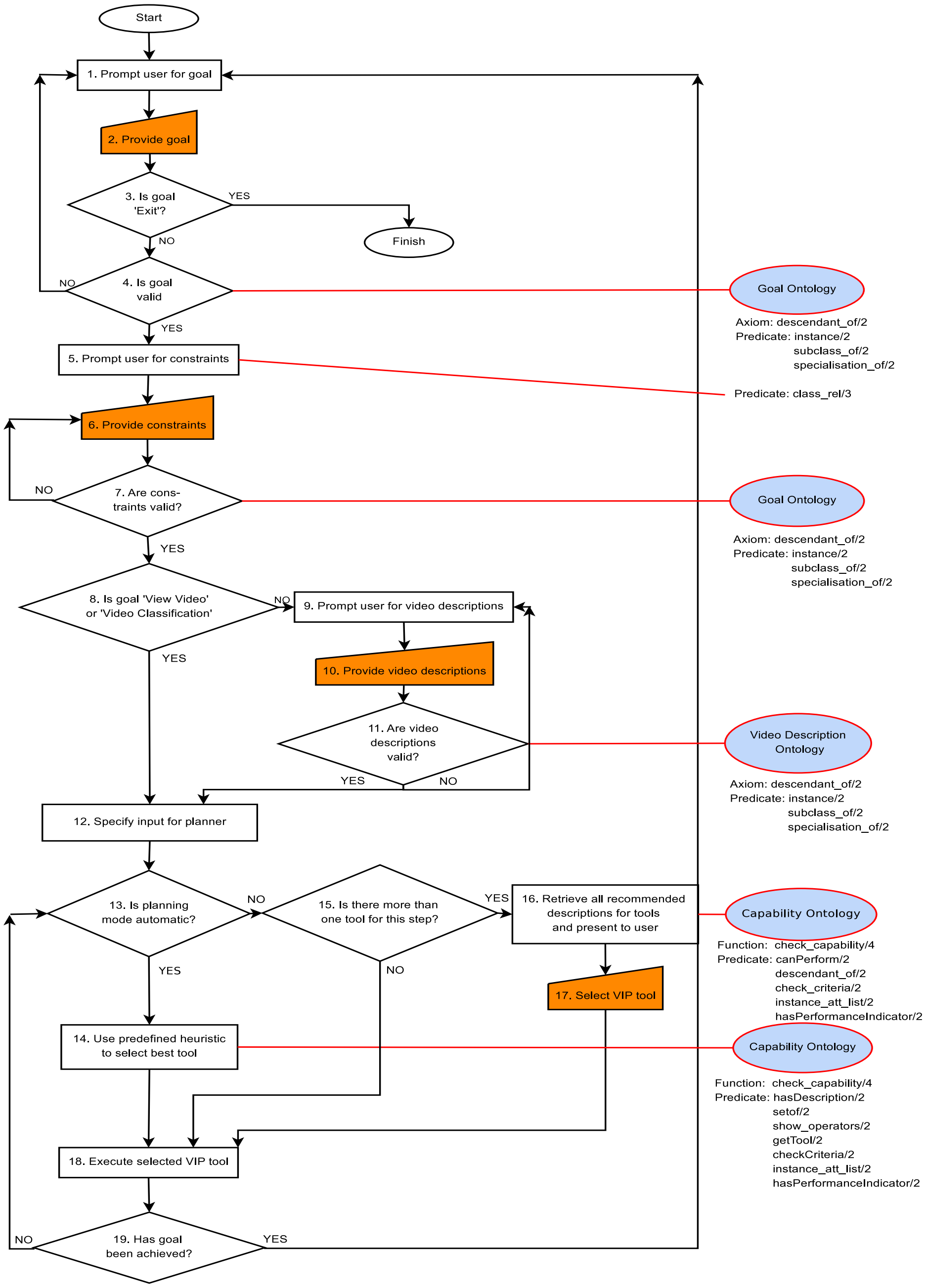

Start
1. Prompt user for goal
2. Provide goal
3. Is goal 'Exit'?
YES
Finish
NO
4. Is goal valid
NO
YES
Goal Ontology
Axiom: descendant_of/2
Predicate: instance/2
subclass_of/2
specialisation_of/2
5. Prompt user for constraints
Predicate: class_rel/3
6. Provide constraints
7. Are cons-traints valid?
NO
YES
Goal Ontology
Axiom: descendant_of/2
Predicate: instance/2
subclass_of/2
specialisation_of/2
8. Is goal 'View Video' or 'Video Classification'
NO
9. Prompt user for video descriptions
YES
10. Provide video descriptions
11. Are video descriptions valid?
YES
NO
Video Description Ontology
Axiom: descendant_of/2
Predicate: instance/2
subclass_of/2
specialisation_of/2
12. Specify input for planner
13. Is planning mode automatic?
NO
15. Is there more than one tool for this step?
YES
16. Retrieve all recommended descriptions for tools and present to user
Capability Ontology
Function: check_capability/4
Predicate: canPerform/2
descendant_of/2
check_criteria/2
instance_att_list/2
hasPerformanceIndicator/2
YES
NO
17. Select VIP tool
14. Use predefined heuristic to select best tool
Capability Ontology
Function: check_capability/4
Predicate: hasDescription/2
setof/2
show_operators/2
getTool/2
checkCriteria/2
instance_att_list/2
hasPerformanceIndicator/2
18. Execute selected VIP tool
19. Has goal been achieved?
NO
YES

Figure 7: Flowchart of the workflow enactor's interaction with the user annotated with ontology use. Shaded processes denote user-provided input.

is checked against the class 'goal' to determine if they belong to the same hierarchy.

Next, the related constraints for the goal are determined (steps 5 and

6). These are additional parameters to specify rules or restrictions that apply to the goal. For the goal 'fish_detection', the relevant constraints are `Performance Criteria, Quality Criteria, Accuracy` and `Occurrence`, contained in the goal ontology. The user may choose to provide all, some or none of these values. Adopting the same principle used for goal retrieval, constraint values are checked if they are valid (step 7), *i.e.* if they are descendants of the class 'Constraint on Goal'.

Then, depending on the goal, the user is prompted for the video descriptions (steps 9 and 10). For the task 'fish_detection', the descriptions required are brightness, clearness and green tone levels. As before, the user may choose to provide all, some or none of these values. The values obtained are checked against the video description ontology (step 11). The checking of validity is again the same as with the goal and constraints, except that the values (instances) are checked against the class 'descriptor' in the video description ontology. In the absence of user information, constraints are set to default values while video descriptions are obtained via preliminary analysis.

Once the goal, constraints and video descriptions are determined, the formulation of the user's problem is complete and this information will be fed to the planner (step 12). The inner workings of the planner will be described in Section 5.1. Basically, the planner seeks to find steps in the form of VIP tools composed in sequential, iterative and conditional fashions in order to solve the task. At each step of the way, the planner attempts to find a suitable tool by consulting the capability ontology (step 14/16). This is done using the 'check_capability' function, described in Algorithm 1 below.

```
check_capability(planning-step S, planning-mode Auto)
If Auto is true
       Retrieve a tool, T that can perform S from capability ontology:
       getTool(S)
       Check that domain-criteria tied to this tool (if any) hold:
       check_criteria(T)
Else
       Display description of S, D to user:
       hasDescription(S, D)
       Retrieve ALL tools that can perform S from capability ontology:
       For each tool ti in Ts
          Retrieve recommended domain descriptions, RD for ti:
          hasPerformanceIndicator(ti, RD)
          Display ti and RD to user
       End for
          Display also system's recommended tool, ts
          getTool(S)
       Prompt user to select a tool T
Return T
```

```
getTool(S)
       canPerform(T, S)
       instance(T, T1)
       descendant_of(T1, tool)
Return T

check_criteria(T)
       If a set of domain-criteria, DC exist for this tool
       hasPerformanceIndicator(T, DC)
          Retrieve the list of preconditions, P for DC:
          instance_att_list(DC, P)
             If all preconditions in P hold
                return DC
             Else return Fail
       Else return 'no_criteria'
```

Algorithm 1: The 'check_capability' function for VIP tool (planning step) selection.

First it retrieves a tool that can perform the planning step. Subsequently, it checks if the selected tool is linked to a list of domain conditions that are deemed suitable for it according to image processing experts' heuristics. Not all tools are tied to domain conditions. The domain conditions that are suitable for this tool are checked against the current domain conditions. If all of them hold, then the tool is selected for execution. Otherwise it will try to find another tool where such conditions hold, failing otherwise. This is applied when the planning mode is automatic (steps 13 and 14). In semi-automatic mode, user will make this tool selection whenever more than one tool is present to perform a planning step. All the tools and their recommended descriptions are displayed to the user who will select one of them (steps 15 and 16). The descriptions are expressed in natural language to ease readability for the user. When a tool is selected, it is applied directly to the video or image in question (step 18). This planning interleaved with execution process continues until the task is solved, *i.e.* when the goal is achieved (step 19).

# 5 Planning and Workflow Enactment

At the heart of the system lies a workflow enactor that interfaces the interactions with the user and coordinates all the activities between the components within the system. The main component is a planner that is responsible for the derivation of video and image processing (VIP) solutions based on the provided goal and domain descriptions. Therefore, the planner is a reasoner that translates the high level non-technical terms (user goals and preferences) to low level technical terms (VIP operations) for workflow composition. This is done with the assistance of the process library and ontologies. Two key innovations of the planner are the ability to support workflow execution (interleaves planning with execution) and can perform in automatic or semi-automatic (interactive) mode. It extends the capabilities of typical planners by guiding users to construct more optimal solutions via the provision of recommended descriptions for the tools. It is also highly adaptable to user preferences.

## 5.1 Planner Design and Representation

The planner was designed with the aim of enhancing the flexibilities of existing Hierarchical Task Network (HTN) planners. HTN planning uses so-called methods or refinements to describe the decomposition of non primitive tasks in terms of primitive and other non primitive tasks. Primitive tasks are directly executable by applying an operator. The planner starts with an abstract network representing the task to be accomplished, and proceeds by expanding the abstract network into more detailed networks lower down in the abstraction hierarchy, until the networks only contain executable actions. HTN methods are a form of domain-specific knowledge (in the form of task decompositions). This greatly reduces the search space of the planner by encoding knowledge on how to go about looking for a plan in a domain and also enables the user to control the type of solutions that are considered. Due to its successes and practicalities, several major HTN planners have been developed over the past decades, namely O-Plan [5], SHOP [25], I-X [30], HyHTN[18] and SHOP2 [24].

The design includes interleaving planning with execution, and providing automatic and semi-automatic (interactive) modes of planning. The semi-automatic mode of planning enables the user to construct more optimal solutions by providing recommended descriptions when more than one tool is available to perform a primitive task.

The implementation of the planner has been kept separate to the modelling of the domain. The planner itself is domain-independent and can be tailored to be used in different problem scenarios as long as the domain descriptions and relevant ontologies are provided. Domain-specific information is encoded in the knowledge base as facts and in the process library as primitive tasks and methods. The planner and domain model are described declaratively using SICStus Prolog 4.0. The planner was built based on ordered task decomposition where the planning algorithm which composes tasks in the same order that they will be executed.

The input to the planner are the goals, objects and conditions of the objects at the beginning of the problem (initial state), and a representation of the actions that can be applied directly to primitive tasks (operators). For HTN planning, a set of *methods* that describe how non primitive tasks are decomposed into primitive and non primitive tasks are also required. The output should be (partial) orderings of operators guaranteed to achieve the goals when applied to the initial state.

A video processing task can be modelled as an HTN planning problem, where a goal list, $G$ is represented as the VIP task(s) to be solved, the primitive tasks, $p$ are represented by the VIP primitive tasks and the operators, $O$ are represented by the VIP tools that may perform the primitive tasks directly. The methods, $M$ specify how the non primitive tasks are decomposed into primitive and non primitive subtasks. The primitive tasks and methods are contained in the process library.

Adapting the conventions provided in Ghallab *et al.* [11], an HTN planning problem is a 5-tuple

$$P = (s_0, G, P, O, M)$$

where $s_0$ is the initial state, $G$ is the goal list, $P$ is a set of primitive tasks, $O$ is a set of operators, and $M$ is a set of HTN methods. A primitive task, $p \in P$ is a 3-tuple

$$p = (\text{name}(p), \text{ preconditions}(p), \text{ postconditions}(p))$$

where name($p$) is a unique name for the primitive task, preconditions($p$) is a set of literals that must hold for the task to be applied and postconditions($p$) is a set of literals that must hold after the task is applied.

An HTN method, $m \in M$ is a 6-tuple

$$m = (\text{name}(m), \text{ task}(m), \text{ preconditions}(m), \text{ decomposition}(m), \text{ effects}(m), \\ \text{postconditions}(m))$$

where name($m$) is a unique name for the method, task($m$) is a non primitive task, preconditions($m$) is a set of literals that must hold for the task to be applied, decomposition($m$) is a set of primitive or non primitive tasks that $m$ can be decomposed into, effects($m$) is a set of literals to be asserted after the method is applied and postconditions($m$) is a set of literals that must hold after the task is applied. The planning domain, $D$, is the pair $(O, M)$.

## 5.2 Primitive Tasks, Operators and Methods

30 VIP operators were identified and implemented for performing the task video classification according to brightness, clearness and algal levels, fish detection and counting. They were developed using a combination of top-down and bottom-up approaches that allowed a suitable level of granularity of the operators [23]. The corresponding primitive tasks that the operators can act upon are encoded in the process library. As stated earlier, primitive tasks are those that can be performed directly by operators or VIP tools.

For each primitive task, its corresponding technical name, preconditions, parameter values, output values, postconditions and effects of applying this task are specified. The preconditions are all the conditions that must hold (prerequisites) for this primitive task to be performed. The effects are conditions that will be asserted or retracted after completion of the task and the postconditions are all the conditions that must hold *after* the task is applied.

Non primitive tasks are decomposable to primitive and non primitive subtasks. Schemes for reducing them are encoded as *methods* in the process library. For each method, the name of the method, the preconditions, decomposition, effects and postconditions are specified. The decomposition is given by a set of subtasks that must be performed in order to achieve this non primitive task. For VIP tasks, the methods are broadly categorised into three distinct types; non recursive, recursive and multiple conditions. Non recursive method is the most common form that does not involve loops or branching of any sort, for example a direct decomposition of a video classification method into processing the image frames, followed by performing the classification on the frames and finally writing the resulting frames onto a video. Recursive methods, as its name suggest, model loops, hence the decomposition of these methods will include itself as a subtask. Multiple conditions arise when there is more than one decomposition for a method. For instance, texture features can be computed by either computing the histogram values followed by computing the statistical moments *or* by computing the Gabor filter. Two separate decompositions will be possible for computing the texture features.

## 5.3 Planner Algorithm

Planning algorithms solve problems by applying *applicable* operators to initial state to create new states, then repeating this process from the new states until the goal state is reached. In HTN planning, the algorithm stops when all the goal tasks have been achieved. The algorithm below describes the main workings of the planner implemented for this thesis.

```
gplanner(initial-state S, goal-list [g1|G], domain D, solution-plan P)
Initialise P to be empty
If goal-list is empty, return P

Consider first element in the goal-list, g1 in goal list [g1|T]
Case 1: If g1 is primitive AND all its preconditions hold
       1.1. If one or more operator instances (tools) match g1
            Retrieve ALL operators (tools) T = [t1,..,tn] from capability
            ontology that can perform g1
            For each tool, ti in T
               Retrieve suitable domain conditions for ti from capability ontology
            End For
              If more than one operator is available to perform g1
                Display all applicable tools with suitable domain conditions
                Prompt user to select preferred tool, tp
              Else tp is the only available operator to perform g1
                Apply tp to S and planning algorithm to rest of goal-list:
                apply-operator(tp, Input_list, Add_list)
                Check that all postconditions of g1 hold
                gplanner(tp(S), G, D, P)
       1.2 Else return fail

Case 2: If g1 is non primitive
       2.1. If a method instance m matches g1 in S
            AND all its preconditions hold
              Append the decompositions of m into the front of the goal-list
              Add all elements in m's Add List to S
              Check that all postconditions of m hold
              Apply planning algorithm to this new list of goals:
              gplanner(S, append(m(g1),G), D, P)
       2.2 Else return fail


% Apply_operator
apply-operator(Tool, Input_list, Add_list, P, S)
    Update solution-plan P with Tool (append Tool to the end of P)
    Execute Tool with parameters Input_list
    Add all elements in Add_list (effects) to S
```

Algorithm 2: Workings of the SWAV planner in semi-automatic mode.

The domain is represented by the predicates that contain video descriptions (*e.g.* brightness, clearness and green tone levels), the constraints (*e.g.* accuracy and processing time), methods (decompositions) and operators (tools to execute the primitive tasks). The algorithm is a recursive function that works on the goal list until it is empty. It inspects each item in the goal list to see if it is a primitive task. If the item is a primitive task, it seeks to find an operator that can perform the primitive task. This is done automatically or semi-automatically, depending on the planning mode selected by the user. Once found, the operator is applied and the primitive task is accomplished. If the task is not primitive, it looks for a method instance that matches it and appends its decompositions to the start of the goal list. The basis for the planning algorithm was taken from HTN planners that generate plans in a totally ordered fashion, where tasks are decomposed from left to right in the same order that they will be executed. In addition, it can plan interactively, interleave planning with workflow execution and has been enriched to make use of knowledge from ontologies.

## 5.4 Automatic or Interactive Mode of Planning

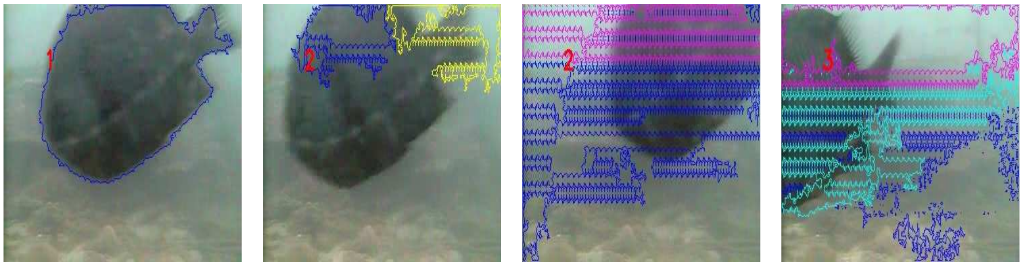
The planner works in either automatic or semi-automatic (interactive) mode, determined by the user before planning takes place. In the automatic mode, user intervention during tool selection is not required. At each planning step, the planner itself selects the tool deemed most optimal based on the domain conditions that match with the tool's recommended domain conditions encoded in the capability ontology. These conditions have been determined based on image processing experts' heuristics (determined empirically). Thus the preferred step in Algorithm 2 is selected by the system rather than the user. When there are no heuristics to guide the tool selection, the first tool encountered that can perform the primitive task is selected for execution. Hence it follows a depth-first style of search.

In the semi-automatic mode, the planning process is **interactive** when more than one tool is available to perform a primitive task. At this level, the planner derives all the applicable VIP tools and their recommended domain descriptions from the capability ontology (See Algorithm 1 in Section 4.5 for the reasoning mechanism of this). The user selects the VIP operator/tool of their choice based on the recommended domain descriptions for each tool as guidance. Thus the planner allows the user to select a tool during the planning process, giving them control and also the ability to make informed decisions. The next section will highlight how this control is followed through with user verification of the final result.

## 5.5 Interleaving Planning with Execution

The planner follows a style of planning interleaved with execution. Once a VIP tool is selected, it is executed before the next planning step is inferred. This is because domain conditions could change as a result of the application of a selected tool. This would then affect the planning process of subsequent steps. However, replanning is not allowed until execution of the whole task is completed. This is due to the fact that intermediate results cannot be used to assess the optimality of the selected VIP tools (neither by the system nor the user) and finding a suitable heuristic for this purpose is not trivial.

Once planning is complete, the user has access to the final video containing the result of applying the tool(s) that they have selected. This will give them a good indication of the optimality of the tool(s) that they have selected. Fig. 8 shows an example of this for a detection task. After viewing this result, they may decide to replan in order to try different choices of tools. Section 6 includes an evaluation of the learnability level achieved by the user in selecting the optimal solutions based on the descriptions provided by the system using the semi-automatic planning mode. The next section will illustrate two examples on how domain descriptions can affect the selection of planning operators.



(a) Adaptive Gaussian Mixture Model. (b) IFD Model. (c) Poisson Model. (d) W4 Model.

Figure 8: Results of applying four different background models for fish detection and counting task for the same video.

# 6 Evaluation

Three hypotheses were formulated by taking into consideration factors such as diversity in user requirements, variety in the quality of the videos (lighting conditions, object movement, *etc.*) and vastness of the data made available:

1. Automated support could be provided for users without image processing expertise to perform VIP tasks in a *time-efficient manner* using a novel semantics- and planning-based workflow composition and execution system without loss of accuracy in the quality of the solutions produced.

2. Constructing VIP solutions using multiple VIP executables employing planning and workflow technologies is more *flexible* and *adaptable* towards changing users' needs than modifying single executable programs.

3. The semantics and planning based automated-assisted mechanism to compose and execute workflows for video processing tasks helps the user *manage and learn* the processes involved in constructing optimal solutions.

## 6.1 Data Set: Ecogrid Videos

Videos are captured, stored and made accessible to marine biologists continuously via Ecogrid [8], a joint effort between the National Center for High Performing Computing (NCHC), Taiwan and several local and international research institutes which provides a Grid-based infrastructure for ecological research. Data is acquired using geographically distributed sensors in various protected sites such as Ken-Ting national park, Hobihu station and Lanyu island. The video streams collected have enabled analysis in underwater reef monitoring, among others. Interesting behaviours and characteristics of marine life such as fish and coral can be extracted from the videos by performing analysis such as classification, detection, counting and tracking.

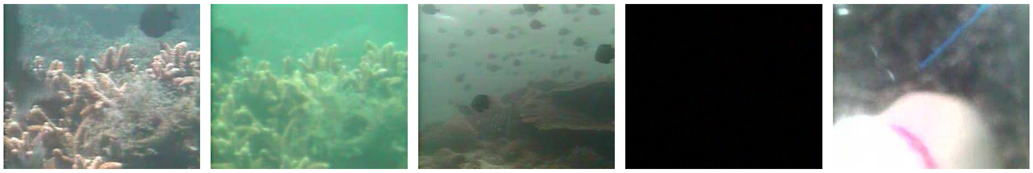


Figure 9: Sample shots from Ecogrid videos. From left to right: clear with fish, algal presence on camera, medium brightness with fish, completely dark and human activity.

As can be seen from the image captures in Fig. 9, the videos were taken in an uncontrolled open sea environment where the degree of luminosity and water flow may vary depending upon the weather and the time of the day. The water may also have varying degrees of clearness and cleanness. In addition, the lighting conditions change very slowly, the camera and the background are fixed, images are degraded by a blocking effect due to the compression, and the fishes are regions whose colours are different from the background and are bigger than a minimal area (they are unusable otherwise). Furthermore, as algae grow rapidly in subtropical waters and on camera lens, it affects the quality of the videos taken. Consequently, different degrees of greenish videos are produced. In order to decrease the algae, frequent and manual cleaning of the lens is required.

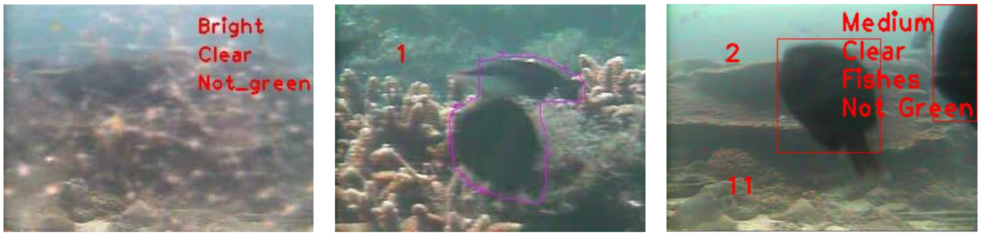
## 6.2 Tasks and Subjects

Based on the video descriptions and other features displayed from the video captures, as well as discussions with marine biologists, several broad categories of tasks have been identified as useful for this test data. These are described below, in order of increasing complexity:

T1. Video classification based on brightness, clearness and algal levels.

T2. Fish detection and counting in individual frames.

T3. Fish detection, counting and tracking in video.



(a) T1: Video classification. (b) T2: Fish detection and counting in frames. (c) T1 & T3: Video classification, fish detection and tracking.

Figure 10: Sample visual results for video processing tasks.

The results are annotated to the original video in text and numerical format. For example, Fig. 10(a) shows the brightness, clearness and green tone values on the resulting video while Fig. 10(b) shows the detected fish and the number of fish in the current frame image on the top left (incorrect in this frame). These tasks can be conducted in a combined fashion for more sophisticated analysis, for instance T1 and T3 can be combined to conduct video classification, fish detection, counting and tracking. Fig. 10(c) shows an example result for this task. The final video is annotated with brightness, clearness, green tone and fish presence values, as well as the number of fish in the current frame (number at the top left of image) and the number of fish so far in the video (number at the bottom left of image).

These tasks will be useful to extract higher level analyses such as population analysis, fish behaviour at certain times of the day or year and suitable environmental conditions for fish presence. However, extracting useful characteristics and features from these videos would take too long to perform manually. One minute's clip requires approximately 15 minutes' human effort on average for basic processing tasks [4]. The following subjects and systems were used for experimental purposes:

S1. An image processing-naive user who constructs the solution with the assistance of the workflow tool using full automatic and/or semi-automatic mode.

S2. An image processing-naive user who solves the goal manually.

S3. An image processing expert who constructs the solution using specialised tools (*e.g.* by writing an OpenCV [15] program).

S4. A workflow modeller who constructs the solution using the workflow tool (*e.g.* by manipulating the process library and ontologies).

## 6.3 Statistical Hypothesis Testing using the *t*-distribution

Experiments demonstrating performance gains of plan generation with the assistance of the workflow tool over manual processing and program generation from scratch were conducted. Performance gains are measured in CPU time, manual processing time, and quality of solutions as compared to the ground truth provided by marine biologists, where appropriate. The experiments were designed according to the principles outlined in [14], where first a hypothesis and its counterpart null hypothesis are formulated, followed by the variables, conditions and subjects for the experiments, then the hypothesis is tested by providing measurement levels to report the results and finally an analysis to accept or reject the null hypothesis. The two sample dependent $t$-test was performed to determine the $t$ value and its corresponding $p$ value in order to accept or reject the null hypothesis. The $t$ value is given by Equation 1 below:

$$t = \frac{\bar{d_e}}{\sqrt{\frac{\sigma_{de}^2}{n}}} \tag{1}$$

where $n$ is the sample size, $\bar{d_e}$ is the mean of the differences between the manual and automatic times and $\sigma_{de}$ is the standard deviation of this mean. Based on the values of $t$ and $n$, a significance level was computed. A significance level of $p < 0.05$ was taken as an acceptable condition to reject the null hypothesis.

Where appropriate, tasks and systems from Section 6.2 will be referred to. All the experiments were conducted on 27 videos selected from Ecogrid. For experiments to test efficiency (Section 6.4) and user learnability (Section 6.6), eight participants from a variety of backgrounds were selected as subjects. None of them possessed image processing expertise; three computer scientists and five non computer scientists. The backgrounds of the non computer scientists include medicine, history and archaeology, physics, ecology and marine biology. The reason for having this variety was to test the usability and effects of the system on a mix of users, with technical expertise (computer scientists) and without, with domain expertise (ecologist and marine biologist) and without. These two experiments also required user-driven evaluation measures. All experiments were conducted on a laptop operating on Ubuntu 8.04 with 512 MB RAM and 1.6 GHz processor speed.

## 6.4 Manual *vs.* Automatic Approaches for Video Classification Task

In this experiment, subjects were asked to classify 10 videos according to brightness, clearness and green tone (algal) levels. First, they were required to conduct the task manually, and then using the workflow tool. Using manual processing, each video was played using a video processing software where the subject may pause and repeat the video in order to determine the brightness, clearness and green tone levels. The subjects record the classification value for brightness as "bright", "medium", or "dark", the classification value for clearness as "clear", "medium" or "blur" and the classification value for green tone level as "green" or "not green". They were advised to select these classification values based on their own judgement.

Using the workflow tool, the subjects selected the option to perform the task 'Video classification according to brightness, clearness and green tone levels'. Then they were prompted to select between a fast and a slow processing mode. For each video, they were required to run the slow processing followed by the fast processing. They were given the freedom to perform this task automatically using just the fast processing mode when they were confident enough to do so, otherwise they would proceed to use the two modes on the videos. The reason for conducting the experiment in this fashion was to test the subjects' confidence in the quality of the fast processing. The number of videos processed before the user switched to using the fast processing mode only was noted.

The fast processing algorithm was taken as the automatic processing time as it has been empirically shown to produce the same quality of results as the slow one. The CPU time taken to perform the task automatically and the time taken to perform the task manually were computed. The accuracy of the results were computed using the following procedure. As there were three classification criteria (brightness, clearness and green tone), each matching value of the automatic (system) or manual (subject) classification value with the ground truth was given a score of 1. For each video manipulated by each subject, a score of 3 would indicate 100% accuracy, in which case all three classification values (brightness, clearness and green tone) matched the recommended values. For all 10 videos manipulated by each subject, an average score was computed. A percentage was then calculated for the quality of the solution produced. The users were also asked three additional questions, whether they noticed any difference in the fast and slow automatic processing times, whether they found the automatic processing less tedious than the manual processing and which method would they prefer to use if the tasks were to be done frequently.

### 6.4.1 Results

Table 1 contains the average time measurements, accuracies of the results and the differences between automatic and manual processing for each subject who participated in the experiment. As explained in the previous section, the metrics were produced based on 10 videos processed by each subject, out of a total of eight subjects. There was a total of 27 videos, where each video was manipulated three times throughout the entire experiment. The classification result provided by a marine biologist was taken as the base line for the accuracy. Based on these values, statistical tests were performed to evaluate the efficiency of the methods of processing and the quality of the solutions produced by the methods.

### 6.4.2 Testing of Efficiency

Statistical hypothesis testing using the $t$-distribution was conducted to measure the dependencies between the results obtained for the times taken to conduct automatic and manual processing. The hypothesis, null hypothesis, independent and dependent variables for this test are given below.

- Hypothesis: Image processing-naive users solve VIP tasks using the workflow tool faster than performing them manually.

- Null hypothesis: There is no difference in the time taken for image processing-naive users to solve the task manually and with the workflow tool.

Table 1: Time and accuracy of automatic versus manual processing, and their differences for video classification task.

| Subject | Automatic | | Manual | | Difference | |
|---|---|---|---|---|---|---|
| | Time (s) | Accuracy (%) | Time (s) | Accuracy (%) | Time $d_e$ | Accuracy $d_a$ |
| 1 | 2.12 | 61.11 | 47.90 | 76.19 | -45.78 | -15.08 |
| 2 | 2.13 | 61.11 | 39.65 | 53.33 | -37.52 | 7.78 |
| 3 | 2.09 | 61.11 | 40.12 | 25.00 | -38.03 | 36.11 |
| 4 | 2.06 | 61.11 | 45.33 | 87.50 | -43.28 | -26.39 |
| 5 | 2.13 | 61.11 | 35.02 | 52.38 | -32.89 | 8.73 |
| 6 | 2.14 | 61.11 | 48.25 | 80.00 | -46.11 | -18.89 |
| 7 | 2.06 | 61.11 | 37.20 | 66.67 | -35.14 | -5.56 |
| 8 | 2.02 | 61.11 | 17.95 | 52.78 | -15.93 | 8.33 |
| Average | 2.09 | 61.11 | 38.93 | 61.73 | -36.83 | -0.62 |

- Independent variables:
  – Data (27 Ecogrid videos of various quality).
  – Subjects and systems used for solving task: S1, S2.
  – Task T1: Classify video according to brightness, clearness and algal levels.

- Dependent variables: Time taken to perform task manually versus time taken to perform task automatically.

The $t$ value, given by equation 1 was computed for this data yielding -11.43. The degree of freedom is set to $n - 1$, which is 7. A value of $t(7)$ = -11.43 corresponds to a significance level of $p \ll 0.0001$. This means that the null hypothesis can be rejected. It can be concluded that the efficiency of automatic processing is significantly higher than the efficiency of manual processing.

### 6.4.3 Testing of Accuracy

A similar statistical hypothesis testing using the $t$-distribution was conducted to measure the dependencies between the accuracies between the results obtained for automatic and manual processing. The hypothesis, null hypothesis, independent and dependent variables for this test are given below.

- Hypothesis: The quality of the solutions produced by image processing-naive users when solving video classification task using the automatic workflow tool is higher than quality of the solutions produced when they perform the task manually.

- Null hypothesis: There is no difference in the quality of the solutions produced by image processing-naive users to solve the task manually and with the workflow tool.

- Independent variables
  – Data (27 Ecogrid videos of various quality).
  – Subjects and systems used for solving task: S1, S2.
  – Task T1: Classify video according to brightness, clearness and algal levels.

- Dependent variables: Quality of results as compared to ground truth.

The $t$ value was computed using Equation 1, yielding -0.09133. The degree of freedom is set to $n-1$, which is 7. A value of $t(7)$ = -0.09133 corresponds to a significance level of $p = 0.4649$. This means that the null hypothesis cannot be rejected. Hence, the accuracy of manual processing, although slightly higher on average, is not significant enough to indicate that it is more accurate than the solutions produced by automatic processing.

### 6.4.4 Analysis

One observation from the results is on the relationship between the manual processing times and accuracy. The accuracy of the subjects' manual classification varied slightly. It was noted that subjects who had domain knowledge (*e.g.* ecologist and marine biologist) had higher levels of accuracies in the video classification than their counterparts without domain expertise (*e.g.* computer scientists). However, they did not take less time in performing this classification. The main findings of this experiment that compares automatic and manual processing using efficiency and accuracy metrics are the following:

- Automatic processing for video classification is on average 94.73% faster than manual processing without loss of accuracy in the solutions produced.

- 75% of the subjects found performing the video processing task using the automatic tool was less tedious than performing it manually.

- All subjects preferred to use the automatic tool over the manual method if they were to conduct the video classification task frequently.

The next experiment was aimed at testing the efficiency and accuracy levels in the workflow tool and conventional image processing approaches when it comes to adapting to changes in user preferences, with a focus on varying domain descriptions.

## 6.5 Single- *vs.* Multiple-Executable Approaches on Software Alteration

This experiment aims to show that the workflow system which adopts a multiple-executable approach adapts quicker to changes in user preferences than its single-executable counterpart (specialised image processing programs). This is the test of adaptability of the workflow system to reconstruct solutions efficiently when the user changes the domain descriptions for a task.

In this experiment, an image processing expert and a workflow modeller have access to the same set of video and image processing tools; the former has an OpenCV program with available image processing algorithms written as functions and the latter in the form of independent executables defined in the process library.

Both subjects were familiar with the systems that they were manipulating. They were given an identical task to perform – fish detection, counting and tracking in a video. Both systems were able to perform this task using a default detecting and tracking algorithm. In the workflow tool, the Gaussian mixture model was defined as the detection algorithm, no methods were defined for the selection of any other detection algorithm. In the OpenCV program, the Gaussian mixture model was used as the detection algorithm. Six scenarios were presented to both subjects containing changes to domain conditions (see Table 2). Both subjects were asked to make modifications or additions of code to their respective systems to cater for these changes in order to solve the VIP task as best as possible. For this purpose, they were both given which detection algorithm should be selected in each case. The number of lines of code (OpenCV for image processing expert and Prolog for workflow modeller) and the time taken to make these modifications were computed for both subjects. A line of code in OpenCV is represented by a valid C++ line of code, *i.e.* a line ending with a semi-colon (`;`), a looping or conditional statement (`if/for/while`). In Prolog, a line of code is a single predicate or fact ending with a full stop (`.`) or a statement ending with a comma (`,`).

The quality of the solutions is calculated as follows. There are two values to be considered, the first is the number of fish in the current frame and the second is the number of fish in the video so far. Each of these is given a score of 1 if there is a match with the ground truth. For each frame, the accuracy could be 0%, 50% or 100%. An average accuracy as a percentage is computed by taking the accuracy of 10 frames (1st, 6th, 11th, ..., 46th) from each video over all 27 videos.

### 6.5.1 Results

Table 2 contains the results obtained for this experiment. For each domain description altered, the time taken to modify the system, the number of lines of code added, and the accuracy of the solution are given.

Table 2: Comparisons of number of new lines of code written, processing times and accuracies of solutions between single-executable image processing program and multiple-executable workflow system to adapt to changing domain descriptions.

| Domain Descriptions | Image Processing Expert | | | Workflow Modeller | | |
|---|---|---|---|---|---|---|
| (User Preference) | New Lines of Code | Time (min.) | Accuracy % | New Lines of Code | Time (min.) | Accuracy % |
| Prefer false alarm than miss | 43 | 16 | 58.25 | 3 | 3 | 59.30 |
| Prefer miss than false alarm | 56 | 23 | 62.55 | 2 | 2 | 64.80 |
| Clear, no background movement | 43 | 16 | 58.46 | 3 | 3 | 60.71 |
| Clear, background movement | 61 | 27 | 60.42 | 2 | 2 | 60.10 |
| Blur, no background movement | 43 | 16 | 60.88 | 3 | 3 | 62.09 |
| Blur, background movement | 57 | 32 | 63.80 | 2 | 2 | 61.22 |
| Average | 50.50 | 21.67 | 60.73 | 2.50 | 2.50 | 61.37 |

The results produced are used to compare the efficiency of two different problem solving systems given equal starting points in the form of available solutions and equal expectations in the alterations required when user preferences change. Despite measuring the lines of code between two programming languages, the experiment does not intend to compare the two programming languages, but rather, to show the differences in effort required (*i.e.* time) to solve video processing problems using two different approaches (single- versus multiple-executable systems).

### 6.5.2 Testing of Efficiency

Statistical hypothesis testing using the *t*-distribution was conducted to measure the dependencies between the results obtained for the times taken to make changes to the workflow tool and OpenCV program. The parameters for this test are given below.

- Hypothesis: Constructing VIP solutions using the workflow tool is faster than modifying existing image processing programs each time a domain description is altered for a fish detection, counting and tracking task.

- Null hypothesis: There is no difference in the time taken to solve the task using the workflow tool and modifying existing image processing programs each time a domain description is altered for the same task.

- Independent variables  
  – Data (27 Ecogrid videos of various quality)  
  – Subjects and systems used for solving task: S3, S4.  
  – Task: T3 with the domain conditions given in Table 2.

- Dependent variables  
  1) Time taken to modify existing OpenCV program versus time taken to encode changes in workflow tool.  
  2) Number of new lines of code added to encode the changes in OpenCV program and workflow tool.

Using the formula provided by Equation 1, the value of $t$ was computed to be 7.01. The degree of freedom is set to $n - 1$, which is 5. A value of $t(5)$ = 7.01 corresponds to a significance level of $p \ll 0.05$. This means that the null hypothesis can be rejected. Thus the workflow tool is faster to adapt to changes in domain descriptions than the image processing program.

### 6.5.3 Testing of Accuracy

Again, statistical hypothesis testing using the $t$-distribution was conducted to measure the dependencies between the results obtained for the accuracies of the solutions provided by the workflow tool and the OpenCV program. The parameters for this test are given below.

- Hypothesis: Constructing VIP solutions using the workflow tool yields more accurate solutions than modifying existing image processing programs each time a domain description is altered for a VIP task.

- Null hypothesis: There is no difference in the quality of the solutions obtained using the workflow tool and modifying existing image processing programs each time a domain description is altered for a VIP task.

- Independent variables  
  – Data (27 Ecogrid videos of various quality)  
  – Subjects and systems used for solving task: S3, S4.  
  – Task: T3 with the domain conditions given in Table 2.

- Dependent variables: Quality of solutions, assessed against manually determined values.

Using the formula provided by Equation 1, the value of $t$ was calculated to be 1.01. The degree of freedom is set to $n - 1$, which is 5. A value of $t(5) = 1.01$ corresponds to a significance level of $p = 0.1794$. This means that the null hypothesis cannot be rejected. Thus, the quality of the solutions produced by the workflow tool, although on average slightly better than the quality of the solutions of the image processing program, is not significant enough to be considered more superior.

### 6.5.4 Analysis

When an existing specialised image processing program can be found to support a specific task, the specialised program works very well. However, when the user requirements (domain descriptions) are altered this is no longer guaranteed without modifications to the program. This modification could range from 16 to 32 minutes for fish detection and counting tasks and take at most 61 lines of code to be written. The workflow tool, however, is adaptable to these changes very efficiently, taking just 3 minutes and 3 lines of code at most. The steps involved for the workflow modeller to encode these changes include adding a method in the process library to encapsulate the new domain descriptions as preconditions, and the relevant detection algorithm. Two more lines are added in the capability ontology to introduce this description as a performance criteria and to tie it to a relevant tool.

In terms of accuracy, the workflow tool on average performed slightly better than the image processing program, however, this difference was not significant enough to conclude that it produced solutions with better quality. Hence, without loss of accuracy, the multiple-executable workflow tool is a more adaptable problem solver than the single-executable image processing program.

## 6.6 User Learnability in Selection of Optimal Tool for Detection Task

In this experiment, the system's ability to help the user learn and manage the processes involved in constructing optimal video processing solutions is tested. An optimal tool is one that yields the best overall performance for the VIP task. When the workflow tool is run in full automatic mode, it self-manages the creation of workflows for the user. This is achieved by making use of expert heuristics in assisting with the planning process. However, the system is not able to assess the optimality of the plan that results from this automatic solution as verifying video processing solutions computationally is not a trivial task. Humans, on the other hand, are able to assess the optimality of the plan by viewing the video containing the results. For example, it is trivial for a human to verify that the system has counted two fish in a frame by observing the count displayed in the resulting frame and the bounding boxes around the fish (e.g. Fig. 10(c)). The aim of this experiment is to test whether the user can compare the results of different tools for the same task and learn the best performing tool from this comparison.

In this experiment, each subject was presented with seven pairs of similar videos and seven pairs of dissimilar videos to perform a fish detection and counting task. Similar videos have the same video descriptions (brightness, clearness and green tone levels), while dissimilar videos have differing video descriptions. For these 14 pairs, it was determined previously that similar videos will use the same detection algorithms (hence same optimal detection tool) while dissimilar videos required different detection algorithms (hence not the same optimal detection tool) and have been used as baseline values for the evaluation. The subject was asked to perform this task using the *semi-automatic* mode of the workflow tool on all 14 pairs of videos. The ordering of the pairs were mixed between similar and dissimilar. The aim was to test if they were able to determine the most optimal tool for the detection algorithm based on the recommended descriptions provided by the workflow tool. If they were, then they should select this tool as the most optimal one in the next similar video presented to them. They should also not conclude to select this tool as the most optimal one in the next dissimilar video presented to them.

Before conducting the experiment, subjects were given a demonstration of one run using the semi-automatic mode to familiarise them with the procedures involved. Furthermore, they had performed the experiment in Section 6.4 and have some familiarity with the system. For each pair, they first conducted the task on the first video given. The workflow tool will display the video to the subject before proceeding to solve the task. Knowing the descriptions of the video (*e.g.* brightness, clearness, movement speed, presence of fish), the subject will have more information when selecting a detection algorithm with the help of the recommended descriptions provided by the workflow tool. After selecting a detection algorithm, the workflow tool will display the final result of this task based on this selection visually. The subject repeated the same task on the same video by trying another detection algorithm, if they wished, and observed the result.

When they were confident with the most optimal tool, they proceeded to perform the same task on the second video in the pair. This video could have similar video descriptions as the previous one (*i.e.* similar) or not (*i.e.* dissimilar). If the video descriptions are similar then the best tool inferred from the previous run should also be the best (most optimal) tool for the second run, otherwise it should not. During the second run, subjects were asked which tool they would choose based on what they have inferred from having conducted the experiment on the first video. Each time they selected the best tool inferred from the first video for the second video in the similar pairs, they were given a score of "correct". Each time they selected the best tool from the first video for the second video in the dissimilar pairs, they were given a score of "incorrect". If they were not able to infer the most optimal tool from the first run, and thus could not infer any tool for the second video based on the first, they were given a score of "incorrect". The subjects were asked five additional questions on the usability of the system. These included verification of the system's design principles, which included the option to provide constraints and video descriptions, usefulness of the recommended descriptions, and ease of use.

### 6.6.1 Testing of User Learnability

Statistical hypothesis testing using the $t$-distribution was conducted to measure the dependencies between the results obtained for the number of times the user selected the correct tool based on a previous similar video and the number of times the user selected the incorrect tool based on a previous dissimilar video. The parameters for this test are given below.

- Hypothesis: The semi-automated mechanism to compose workflows for fish detection and counting task helps the user manage and learn the processes involved in constructing optimal solutions.

- Null hypothesis: The descriptions provided by the workflow tool using the semi-automated mechanism to compose workflows for fish detection and counting task do not assist the user to learn the optimal solutions.

- Independent variables
  – Data: 14 pairs of videos from 27 Ecogrid videos, 7 similar pairs and 7 dissimilar pairs.
  – Subjects and systems used for solving task: S1.
  – Task: T2 Detect and count fish in frames.

- Dependent variables
  – Number of times correct tool was selected as optimal tool in second video for similar pairs of videos.
  – Number of times incorrect tool was selected as optimal tool in second video for dissimilar pairs of videos.

### 6.6.2 Results

Table 3 shows the results obtained to assess the level of user learnability when using the workflow tool for seven pairs of similar videos and seven pairs of dissimilar videos.

Table 3: Number of times "correct" tool was selected in seven similar pairs of videos, number of times "incorrect" tool was selected in seven dissimilar pairs of videos.

| Subject | Similar Pairs<br>No. correct choices<br>$c$ | Dissimilar Pairs<br>No. incorrect choices<br>$i$ | Difference<br>$c - i$<br>$d$ |
|---|---|---|---|
| 1 | 6 | 2 | 4 |
| 2 | 4 | 3 | 1 |
| 3 | 5 | 2 | 3 |
| 4 | 6 | 1 | 5 |
| 5 | 4 | 3 | 1 |
| 6 | 6 | 2 | 4 |
| 7 | 4 | 2 | 2 |
| 8 | 5 | 3 | 2 |
| Average | 5 | 2.25 | 2.75 |

The value of $t$ was calculated to be 5.59 using Equation 1.The degree of freedom is set to $n - 1$, which is 7. A value of $t(7) = 5.59$ corresponds to a significance level of $p = 0.0004$. This means that the null hypothesis can be rejected. With this, it can be concluded that subjects, more often than not, selected the correct (optimal) tool when they were presented with a similar video and did not choose this tool by chance. Hence, the semi-automatic mode has helped the user learn and manage the processes involved in selecting the optimal steps when solving a task.

### 6.6.3 Analysis

The results in Table 3 indicate that subjects were able to select the correct optimal tool when they were presented with a similar video 5 out of 7 times on average (71.43% of the time). This is relatively high compared to the instances when they incorrectly chose an optimal tool learnt from a video for another video that is not similar to the one where they have inferred the tool from, which was 2.25 out of 7 times on average (32.14% of the time). The statistical test proves that the workflow tool is indeed able to help subjects learn optimal VIP tools without having any image processing knowledge, but purely from their own visual capabilities in judging visual descriptions and from the textual descriptions provided by the system.

The user's understanding is tested via the provision of the description of the tools when there was more than one tool to perform a task. By repeating the same task using a different tool, the user can gain insight into how a different solution can be produced and could then 'judge' which tool should be used for producing the most optimal solution. Hence when a new video is presented and the same task is performed, the user would have learnt and gained confidence in selecting the most suitable tool. The user can evaluate the performance of any particular tool that they have selected for a particular task. Using this feedback loop, the system can now learn which tools work better for which type of videos (according to their video descriptions) and under which constraint conditions.

To further validate the system's usability and learnability features, the findings of the questionnaire are summarised as follows:

- All subjects agreed on the appropriateness of having the option to provide constraints and video descriptions to specify the VIP task in more detail.

- All subjects found the recommended domain descriptions suitable and helpful in assisting them make more informed decisions when selecting the best detection algorithms (*i.e.* VIP tools).

- All subjects felt it was appropriate to be given the control to select tools despite not having image processing expertise.

- On average, subjects were able to make the decision to run the fast processing option for video classification task after just 6.4 runs.

- All subjects would prefer to use the automated tool if they were to perform video classification and fish detection and counting tasks frequently.

These findings indicate that although the workflow tool and the domain that it manipulates are both complex, users without expertise in workflow or image processing domains can learn how to use the workflow tool and they can even learn some aspects of solving image processing problems via selection of VIP tools. These were achieved in a short period of time as indicated by the experiment's results. The experimental findings also verify and validate the strength of the integrated approach. The efficiency and accuracy of the results obtained indicate that the workflow system is able to produce results comparable to those produced by humans and by competing systems (*e.g.* image processing programs). The use of ontologies has been proven to be beneficial via the provision of recommended descriptions that were useful to users. The planner's correctness and completeness has been tested on a set of VIP tasks (goals), constraints and detection algorithms.

# 7 Conclusions

This research has contributed towards the provision of automatic workflow composition via a novel framework that hinges on two key technologies, ontologies and planning. The integration has resulted in a semantics-rich and flexible mechanism that is beneficial for the scientific, research and application communities.

The framework has been evaluated on video processing tasks, in particular for underwater videos that vary in quality and for user requirements that change. Higher time-efficiency has been achieved in automated-supported video processing tasks without loss of accuracy. In addition, a new, more flexible approach for video processing which utilises multiple reusable executables was introduced. Consequently, non-experts can also learn to construct optimal video processing workflows which was impossible prior to this.

Optimistically, this framework could eventually be used for general-purpose video processing, not just confined to underwater videos. It could also serve as an educational tool for naive users on a larger scale. At present, this approach is being integrated onto a heterogeneous multi-core environment via a resource scheduler and also to a web-based front-end. It is envisaged that this workflow will be extended to support the management of live video streams for on-demand user queries.

# References

[1] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. *Business Process Execution Language for Web Services Version 1.1 (BPEL).* IBM, BEA Systems, Microsoft, SAP AG, Siebel Systems, 2003. `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf`. Last accessed: Apr 22 2013.

[2] J. Blythe, E. Deelman, and Y. Gil. Automatically Composed Workflows for Grid Environments. *IEEE Intelligent Systems*, 19(4):16–23, 2004.

[3] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Managing the Evolution of Dataflows with VisTrails. In *IEEE Workshop*

*on Workflow and Data Flow for Scientific Applications (Sciflow'06)*, pages 71–75, 2006.

[4] Y. H. Chen-Burger and F. P. Lin. A Semantic-based Workflow Choreography for Integrated Sensing and Processing. In *The 9th IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA'05)*, 2005.

[5] K. Currie and A. Tate. O-Plan: the Open Planning Architecture. *Artificial Intelligence*, 52(49-86), 1991.

[6] E. Deelman, D. Gannon, M. Shields, and I. Taylor. Workflows and e-Science: An Overview of Workflow System Features and Capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.

[7] E. Deelman, G. Singh, M.H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, K. Vahi, B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems. *Scientific Programming Journal*, 13(3):219–237, 2005.

[8] Ecogrid. National Center for High Performance Computing, Hsin-Chu, Taiwan, 2006. `http://ecogrid.nchc.org.tw`. Last accessed: Apr 21 2012.

[9] I. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2nd edition, 2003.

[10] I. Foster, J. Voeckler, M. Wilde, and Y. Zhao. Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation. In *14th Conference on Scientific and Statistical Database Management*, pages 37–46, 2002.

[11] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., 2004.

[12] Y. Gil, V. Ratnakar, E. Deelman, G. Mehta, and J. Kim. Wings for Pegasus: Creating Large-scale Scientific Applications using Semantic Representations of Computational Workflows. In *Proceedings of the 19th National Conference on Innovative Applications of Artificial Intelligence (IAAI'07)*, pages 1767–1774. AAAI Press, 2007.

[13] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Springer, 2004.

[14] D. C. Howell. *Statistical Methods for Psychology*. Belmont, CA, 6th edition, 2007.

[15] Intel. *Open Source Computer Vision (OpenCV) Library.* 2006. `http://sourceforge.net/projects/opencvlibrary`. Last accessed: Apr 22 2013.

[16] J. Kim, M. Spraragen, and Y. Gil. An Intelligent Assistant for Interactive Workflow Composition. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI'04)*, pages 125–131. ACM Press, 2004.

[17] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice & Experience*, 18(10):1039–1065, 2005.

[18] T. L. McCluskey, D. Liu, and R. M. Simpson. GIPO II: HTN Planning in a Tool-supported Knowledge Engineering Environment. In *ICAPS'03*, pages 92–101, 2003.

[19] G. Nadarajan and Y. H. Chen-Burger. Goal, Video Descriptions and Capability Ontologies for Fish4Knowledge Domain. In *Special Session on Intelligent Workflow, Cloud Computing and Systems, (KES-AMSTA'12)*, 2012.

[20] G. Nadarajan, Y. H. Chen-Burger, and R. B. Fisher. A Knowledge-Based Planner for Processing Unconstrained Underwater Videos. In *IJCAI'09 Workshop on Learning Structural Knowledge From Observations (STRUCK'09)*, 2009.

[21] G. Nadarajan, Y. H. Chen-Burger, and R. B. Fisher. SWAV: Semantics-based Workflows for Automatic Video Analysis. In *Special Session on Intelligent Workflow, Cloud Computing and Systems, (KES-AMSTA'11)*, 2011.

[22] G. Nadarajan and A. Renouf. A Modular Approach for Automating Video Processing. In *12th International Conference on Computer Analysis of Images and Patterns (CAIP'07)*, 2007.

[23] G. Nadarajan, C. Spampinato, Y. H. Chen-Burger, and R. B. Fisher. A Flexible System for Automated Composition of Intelligent Video Analysis. In *7th International Symposium on Image and Signal Processing and Analysis (ISPA'11)*, 2011.

[24] D. Nau, T. C. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu, and F.Yaman. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[25] D. S. Nau, Y. Cao, A. Lotem, and H. Muñoz Avila. SHOP: Simple Hierarchical Ordered Planner. In *International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 968–973, 1999.

[26] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[27] T. Oinn, M. Greenwood, M. Addis, N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe. Taverna: Lessons in Creating a Workflow Environment for the Life Sciences. *Concurrency and Computation: Practice and Experience*, 18(10):1067–1100, 2006.

[28] A. Renouf and R. Clouard. Hermes - A Human-Machine Interface for the Formulation of Image Processing Applications, 2007. `https://clouard.users.greyc.fr/Hermes/index.html`. Last accessed: April 22 2013.

[29] T. Tannenbaum, D. Wright, K. Miller, and M. Livny. Condor – A Distributed Job Scheduler. In T. Sterling, editor, *Beowulf Cluster Computing with Linux*. MIT Press.

[30] A. Tate. Intelligible AI Planning - Generating Plans Represented as a Set of Constraints. In *Proceedings of the Twentieth British Computer Society Special Group on Expert Systems International Conference on Knowledge Based Systems and Applied Artificial Intelligence (ES'00)*, pages 3–16. Springer, 2000.

[31] I. Taylor, E. Deelman, D. Gannon, and M. Shields. *Workflows for e-Science*. Springer, New York, 2007.

[32] I. Taylor, M. Shields, I. Wang, and A. Harrison. The Triana Workflow Environment: Architecture and Applications. In I. Taylor, E. Deelman, D. Gannon, and M. Shields, editors, *Workflows for e-Science*, pages 320–339. Springer, New York, 2007.

[33] I. Taylor, M. Shields, I. Wang, and O. Rana. Triana Applications within Grid Computing and Peer to Peer Environments. *Journal of Grid Computing*, (2):199–217.

[34] S. van Splunter, F. M. T. Brazier, J. A. Padget, and O. F. Rana. Dynamic Service Reconfiguration and Enactment using an Open Matching Architecture. In *International Conference on Agents and Artificial Intelligence (ICAART'09)*, pages 533–539, 2009.

[35] G. von Laszewski and M. Hategan. Java CoG Kit Karajan/Gridant Workflow Guide. Technical report, Argonne National Laboratory, Argonne, IL, USA, 2005.