Intelligent Visual Reasoning Tutor

Eric Wang
wang@me.skku.ac.kr

Yong Se Kim
yskim@skku.edu

Creative Design and Intelligent Tutoring Systems Research Center
Sungkyunkwan University, Suwon, Korea

Abstract

Visual reasoning is an essential skill for many disciplines in engineering and architecture. We describe an intelligent tutoring system for visual reasoning that uses the missing view problem, a learning contents model based on skills, lessons, and problems, and a learner model that measures domain competence as a set of skills. Learning contents and pedagogical teaching strategy are stored in ontologies, which can be customized by the teacher.

1. Introduction

The ability to visualize and reason about geometric aspects of 3D objects is critical for success in engineering activities. Visual reasoning capability is becoming more significant as the functionality and the usage of computer-aided engineering systems increases, especially since 3D objects must be understood in the 2D space of the computer screen. Yet the instruction of visual and spatial reasoning skills presents challenges for traditional classroom instruction methods. There is an emerging need for intelligent instructional tools that can assist learners in the development of these skills.

We describe the development of the Intelligent Visual Reasoning Tutor (IVRT), an intelligent tutoring system for visual reasoning that can adaptively support different learners' needs, track learners' progress, and provide active critiquing. IVRT provides a learning system with which a learner can develop visualization and spatial reasoning capabilities in a self-paced series of exercises. IVRT is targeted toward freshmen undergraduate students in all engineering disciplines, to supplement a one-semester course.

The Intelligent Visual Reasoning Tutor uses the *missing view problem*, in which two consistent, principal orthographic projections are given, and the objective is to provide the third orthographic projection such that the three views correspond to a valid 3-D solid object [1]. An example of a missing view problem is shown in Figure 1. This type of problem requires the application of visual analysis and visual synthesis [2], which builds the foundations of the visual reasoning processes.

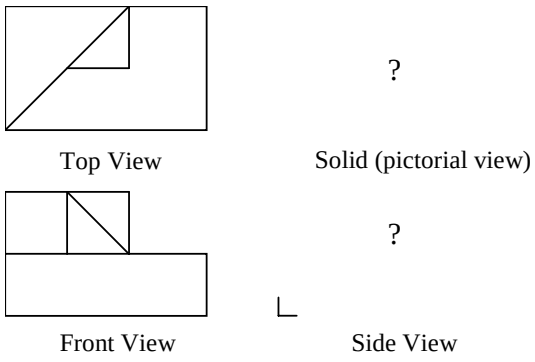
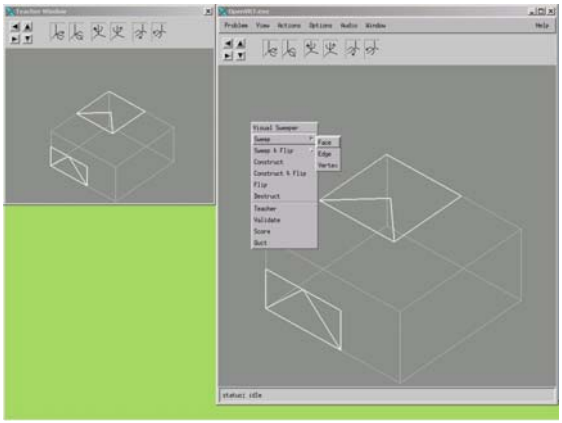


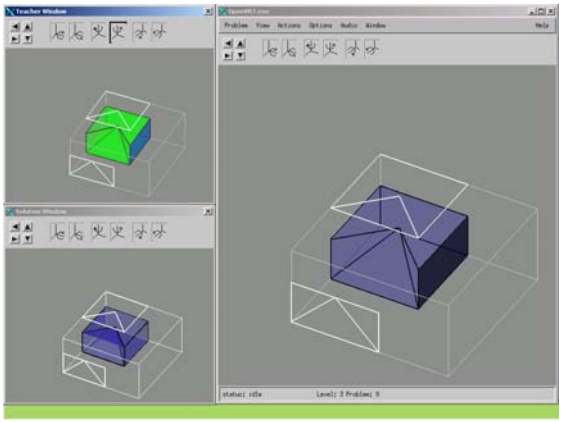
Figure 1. Example of a missing view problem

2. Visual Reasoning Tutor

We have previously developed an instructional software system called Visual Reasoning Tutor (VRT) [3][4]. Two main components of the VRT system have been integrated into the current IVRT system: the Visual Sweeper module, which provides geometric and graphics operations for constructing solid objects from orthographic projections, and the Visual Teacher module, which provides adaptive evaluation and critiquing. These components are shown in Figure 2, running in embedded mode within the current IVRT system.



(a) Visual Sweeper command menus



(b) Teacher and Solution windows

Figure 2. Visual Sweeper and Visual Teacher modules

2.1 Visual Sweeper

The Visual Sweeper, shown in the large window on the right in Figure 2(a,b), provides interactive *sweeping operations* [5], which are the inverse operation of orthographic projection as applied to each face. Sweeping operations are an intuitive, visually-oriented operation to construct 3D solids from orthographic projections. The learner selects a sequence of edges in one orthographic view to form a loop. The Visual Sweeper constructs a 2D face whose boundary is this loop, then allows it to be swept in 3D space by dragging the mouse, such that the projection of the swept face is always consistent with its originating orthographic view. Then by visually analyzing the projection of the swept face in the other orthographic view, the learner interactively positions the swept face to satisfy both views. In this way, the learner incrementally constructs a solution solid through a sequence of sweeping operations.

Variations of the sweeping operation include *face sweep*, in which all vertices are swept; *edge sweep*, where one specified vertex remains fixed, and one specified edge is swept (which implies that the face "stretches" as necessary to remain consistent with the originating view), and *vertex sweep*, in which one specified edge remains fixed, and a specified vertex is translated. The edge and vertex sweep operations allow the construction of slanted faces that are not orthogonal to either orthographic view.

2.2 Visual Teacher

The Visual Teacher module captures and critiques the learner's reasoning. It incorporates problem solving knowledge for missing view visual reasoning as a set of CLIPS rules [6]. It provides the following services: evaluation of a partial solution solid, hinting for the next face to be manipulated, display of the solution solid, and calculation of the learner's score, on a 100-point scale.

- The Teacher window, shown in the upper left of Figure 2(b), provides on-demand evaluation of a learner's solution solid, by color-coding all faces of the learner's solution as correct (green), partially correct (yellow), or incorrect (orange). A partially correct face is one that is not correct yet, but could be made correct with additional operations.

- The Teacher window can also display a hint for the next face to be created, by showing that face in blue color. However, as students have shown a tendency to overuse this mechanism, it is now password-protected.

- The Solution window, shown in the lower left of Figure 2(b), displays the solution solid. A given missing view problem may have multiple valid 3D solution solids. For each problem, we pre-compute all possible solution solids, and store these as part of the problem data. At run-time, the Teacher automatically determines the solution solid that is closest to the learner's current solid.

- The Teacher calculates the learner's score for a problem from the ratio of correct faces to all faces of the closest solution solid, and from penalties incurred for certain commands that indicate non-ideal solution sequences.

3. Intelligent Visual Reasoning Tutor

We have developed a successor system called Intelligent Visual Reasoning Tutor (IVRT), which embeds VRT within an intelligent tutoring system framework. IVRT uses learning contents consisting of *skills, lessons, and problems*, and a learner model that records learners' skill scores and activity history.

- A *lesson* is a text or multimedia resource that teaches the core concepts for missing view visual reasoning, including the orthographic projection of a solid onto viewing planes, and the inverse operation of converting the orthographic projections back to 3-D solid faces.

- A *problem* is an instance of a missing view problem, consisting of two orthogonal views. The learner's objective is to create a valid 3D object that is consistent with both views, using the Visual Sweeper module.

- A *skill* is a domain concept or problem-solving process, which has been identified *a priori* by the teacher or domain expert as having significant pedagogical value, requiring explicit instruction and measurable expertise. A set of 15 skills has been identified for the missing view visual reasoning domain, such as *multiple loop* for properly handling orthographic views with multiple loops, *face sweep* for properly applying the face sweep command, etc.

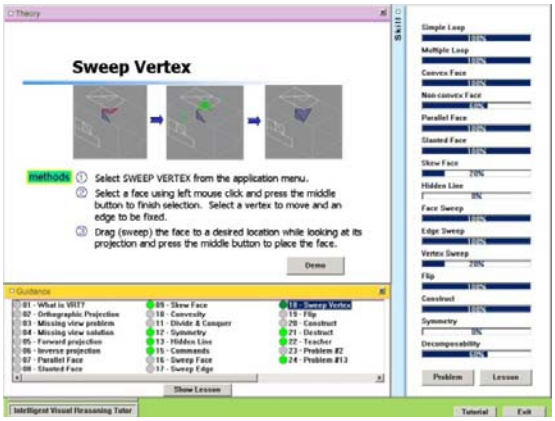
A learner's domain competence is measured by the learner's skill levels, as a set of scores in the interval [0, 100], plus additional data reflecting the learner's command history within the Visual Sweeper module. A new learner begins at 0 score in every skill. As the learner solves problems, the learner earns skill points. This causes more advanced lessons and problems to become available for selection. The learner's objective is to increase every skill score to 100, which completes the tutorial.

3.1 Learner's Interface

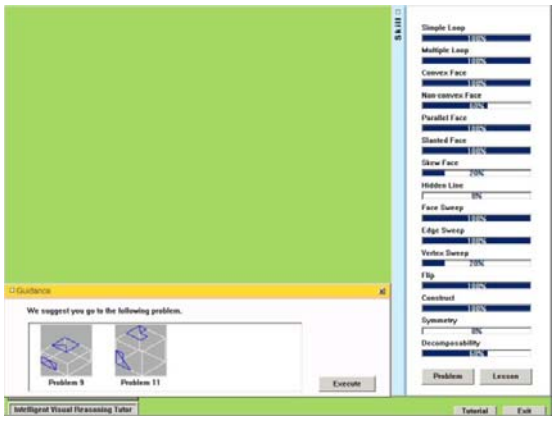
The Learner's Interface provides skill level display as a set of *skill bars*, and interactive selection of lessons and problems based on the learner's current skill levels, as shown in Figure 3.

- When the learner requests the current set of lessons, a comprehensive list of all lessons is presented, as shown at the bottom of Figure 3(a). Within this list, lessons are color-coded to indicate the learner's mastery. Green dots indicate relevant lessons based on the learner's current skill scores, gray dots indicate lessons that the learner has already mastered, and white dots represent advanced lessons whose requirements the learner has not yet fulfilled. For convenience, the comprehensive list of all lessons is always shown, which allows learners to consult earlier lessons at any time.

- The problem selection window, in Figure 3(b), shows the subset of available problems based on the learner's current skill scores. The learner may choose any problem within this subset, which launches the Visual Sweeper module.



(a) Lesson selection and display



(b) Problem selection

Figure 3. Learner's Interface with skill bars, lesson guidance, and problem selection

4. Customization of Learning Contents

IVRT's learning contents and learner model are formalized as ontologies. Pedagogical teaching strategy is represented as inference rules, which are executed by a separate inference engine. We use Protégé with OWL plugin for ontology editing, and Jess [7] as the rule inferencing engine, with XSLT conversion from OWL to Jess.

The teacher can customize the learning contents of skills, lessons, and problems by editing the ontology files.

- *Skills* comprise the link between lessons and problems. Skills may be arranged in a hierarchy of prerequisite skills, to impose an ordering on the sequence of instruction.

- Each *lesson* is associated with the skill(s) to which it contributes, and their eligible score ranges, which determines whether the lesson is to be presented to a learner. The domain expert usually defines at least one lesson for each skill.

- Each *problem* is also associated with the skill(s) that it requires, and their required score ranges and reward values. When the learner solves a problem successfully, the problem's reward values for its associated skills are added to the learner's skill scores.

As the learner reads lessons, solves problems, and earns increases in skill scores, new lessons and problems will be "activated", while old lessons and problems may be considered to be "mastered", and are suppressed. In this way, the domain expert can define a natural order for the presentation of the course material, while still allowing a flexible, customized response based on each learner's individual progress.

Additionally, the teacher can customize the pedagogical teaching strategy, which determines the presentation sequence of lessons and problems based on a learner's skill scores and other history data. IVRT's pedagogical strategy is implemented as a set of inference rules, which are executed in Jess. To assist non-programmers in writing inference rules without requiring Jess expertise, we have developed a prototype rule editor, shown in Figure 4, with a simplified syntax, high-level language abstractions, and automatic conversion to Jess. Using this interface, the teacher can edit any of the pedagogical teaching rules at run-time, and immediately evaluate its effects.

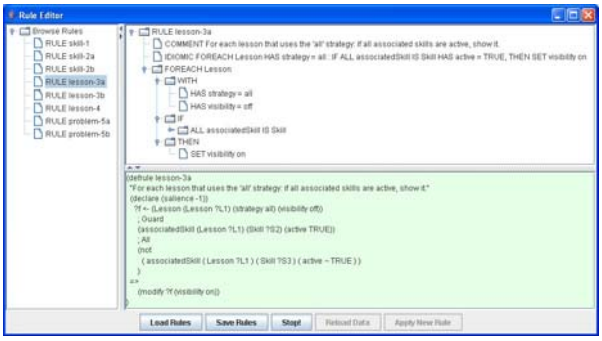


Figure 4. Rule editor with conversion to Jess

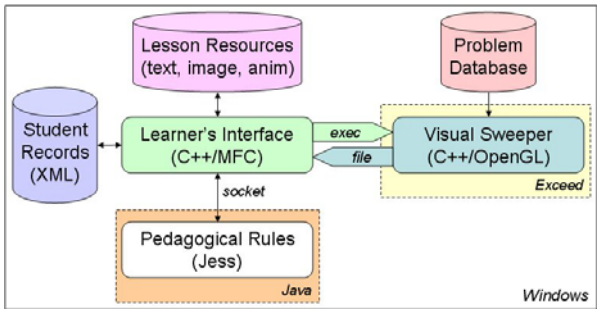


Figure 5. Application architecture of IVRT

5. Application Architecture of IVRT

IVRT is implemented as a set of applications, which communicate at run-time through various mechanisms, as shown in Figure 5.

- The Learner's Interface is a native C++ application for Windows. It provides text and graphics display using standard Microsoft Foundation Classes (MFC) controls, and animation display by invoking the standard Windows file 'open' command, which Windows automatically delegates to a separate utility application that is associated with the animation file's extension type, e.g. Windows Media Player or RealPlayer.

- The Visual Sweeper/Visual Teacher module is a C++ application for X Windows and OpenGL. It executes within Hummingbird Exceed, an X Windows emulation library for Windows. The Learner's Interface executes the Visual Sweeper application as a child process, passing the problem data, and it retrieves the learner's score and command usage data on completion.

- Pedagogical rules that implement the pedagogical teaching strategy are implemented in Jess, as a set

of about 20 Jess rules. On start-up, the Learner's Interface executes Jess within a Java VM as a child process. Thereafter, both of these modules run concurrently, and communicate through sockets, using a simple text protocol. The socket-based communication is inherently asynchronous (non-blocking), so a synchronous version has been developed on top of it. This allows the Learner's Interface's C++ code to abstract away the details of communicating with the Jess rule engine, and treat it as if it were a library of functions callable directly from C++.

- Learners' records are stored as XML files. When a learner logs into IVRT, the Learner's Interface loads the learner's record, displays its skill scores in the skill bars, and also sends the learner's skill scores and other data to the Pedagogical Rules module, which automatically calculates the learner's available lessons and problems. When the learner requests the list of lessons (problems), the Learner's Interface queries the Rules module for the current list, then displays them. Selecting a problem causes the Visual Sweeper to be executed. The result of the Visual Sweeper session is sent to the Pedagogical Rules module, which updates the learner's skill scores, lessons, and problems, and these are written to the learner's XML record.

6. Conclusion and Future Work

IVRT is an intelligent tutoring system for visual reasoning, suitable for use at the undergraduate level. It is also a framework for studying the teacher's customization of learning contents and pedagogical teaching strategy. Preliminary results indicate that ontology-based representation and editing of learning contents can provide rich customization capabilities. Future work includes (1) assessing the ongoing need for dedicated "smart" editing tools, such as the rule editor, (2) enhanced logging of the learner's command history, and (3) use of data mining to identify the learner's preferences and learning styles.

References

[1] Wilde, D. J., "The geometry of spatial visualization: two problems", *8th IFTOM World Congress*, Prague, Aug. 1991.

[2] Kim, Y. S., Astley, M., Pariente, F., and Zhao, H., "Instructional software development for visual reasoning: the first phase", *Proc. Int'l. Conf. on Engineering Computer Graphics and Descriptive Geometry*, Tokyo, 1994.

[3] Kim. Y. S., Moon, C., Chauhan, S., Hubbard, C., Mengshoel, O., and Zhao, H., "Visual Reasoning Tutor (VRT): Instructional Software System for Missing View Problem", *Proc. ASEE Engineering Design Graphics Division Conf.*, Ames, IA, Nov. 1995.

[4] Hubbard, C., Mengshoel, O., Moon, C., and Kim, Y. S., "Visual reasoning instructional software system", *Computers and Education*, Vol. 28, No. 4, pp. 237–250, 1997.

[5] Zhao, H., and Kim, Y. S., "Geometric Operations for Visual Reasoning of a Solid from Orthographic Projections", *Advances in Engineering Software*, Vol.30, No.7, 1999.

[6] Mengshoel, O. J., Chauhan, S., and Kim, Y. S., "Intelligent Critiquing and Tutoring of Spatial Reasoning Skills", *AI in Engineering Design, Analysis, Manufacturing*, Vol.10, 1996.

[7] Friedmann-Hill, E., *Jess in Action*, Manning, 2003.