Spring 2015

VRCC-3D+: Qualitative spatial and temporal reasoning in 3 dimensions

Nathan Eloe

VRCC-3D+: QUALITATIVE SPATIAL AND TEMPORAL REASONING IN 3 DIMENSIONS

by

NATHAN WEBSTER ELOE

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2015

Approved by

Jennifer L. Leopold, Advisor
Chaman L. Sabharwal, Co-Advisor
Wei Jiang
Zhaozheng Yin
Matt Insall

PUBLICATION DISSERTATION OPTION

This dissertation consists of the following five articles that have been published or submitted for publication as follows:

Pages 16–27 have been published in the Proceedings of the 18th International Conference on Distributed Multimedia Systems (DMS 2012).

Pages 28–48 have been published in the Proceedings of the IEEE Symposium on Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP 2013).

Pages 49–67 have been submitted to the International Journal of Software Engineering and Knowledge Engineering (IJSEKE), and is an invited extension of a paper published in the Proceedings of the 19th International Conference on Distributed Multimedia Systems (DMS 2013).

Pages 68–89 have been accepted to the Journal of Visual Languages and Computing (JVLC), and was originally published in the Proceedings of the 20th International Conference on Distributed Multimedia Systems (DMS2014).

Pages 90–107 have been accepted for publication in in Polibits, and was originally accepted at the Mexican International Conference on Artificial Intelligence 2014 (MICAI 2014).

Papers have only been modified to allow for the Missouri S&T formatting guidelines for dissertations. Some tables and figures have been split to account for increased margin size. The content of these papers remains unchanged.

ABSTRACT

Qualitative Spatial Reasoning (QSR) has varying applications in Geographic Information Systems (GIS), visual programming language semantics, and digital image analysis [17, 53]. Systems for spatial reasoning over a set of objects have evolved in both expressive power and complexity, but implementations or usages of these systems are not common. This is partially due to the computational complexity of the operations required by the reasoner to make informed decisions about its surroundings. These theoretical systems are designed to focus on certain criteria, including efficiency of computation, ease of human comprehension, and expressive power. Sadly, the implementation of these systems is frequently left as an exercise for the reader.

Herein, a new QSR system, VRCC-3D+, is proposed that strives to maximize expressive power while minimizing the complexity of reasoning and computational cost of using the system. This system is an evolution of RCC-3D; the system and implementation are constantly being refined to handle the complexities of the reasoning being performed. The refinements contribute to the accuracy, correctness, and speed of the implementation. To improve the accuracy and correctness of the implementation, a way to dynamically change error tolerance in the system to more accurately reflect what the user sees is designed. A method that improves the speed of determining spatial relationships between objects by using composition tables and decision trees is introduced, and improvements to the system itself are recommended; by streamlining the relation set and enforcing strict rules for the precision of the predicates that determine the relationships between objects. A potential use case and prototype implementation is introduced to further motivate the need for implementations of QSR systems, and show that their use is not precluded by computational complexity.

ACKNOWLEDGMENTS

Thanks to my advisors and my committee for leading me through this process as painlessly as possible and to my friends and family for helping me keep it together when the "as possible" caveat kicked in.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

LIST OF TABLES

PAPER V

1. INTRODUCTION

Qualitative Spatial Reasoning (QSR) has varying applications in Geographic Information Systems (GIS), visual programming language semantics, robotics, and digital image analysis [17, 53]. Systems for spatial reasoning over a set of objects have evolved in both expressive power and complexity. The design of each system focuses on certain criteria, including efficiency of computation, ease of human comprehension, and expressive power.

Spatial reasoning is a task that humans perform constantly; understanding the structure of one's surroundings is key in being able to navigate, interract, and exist. Understanding basic principles such as object permanence, or an intuitive understanding of the basic tenants of physics allow humans to make decisions about potentially life-threatening events. For example, imagine security camera footage showing a person carrying a bag, but after walking behind a pillar (or other obstruction) and emerging, is no longer carrying the bag. A human being watching the footage might become suspicious, and take appropriate action. However, the information used by the person taking action was not only supplied by the footage; humans understand that the person who walked from behind the obstruction is indeed the same person who originally passed behind it. Also, they know the bag must still exist; as such the bag could be somewhere behind the obstruction. The observer knows all of this information through experience and observation of the basic tenants of physics, not through rigorous calcluation of obscuration and connection.

Integration of QSR with other information is not an easy task. A large number of existing QSR systems have limitations that inhibit their usage. The majority of the earlier systems such as the Region Connection Calculi (RCC) formulations are

limited to two dimensions. Furthermore, few if any of the systems have implementations; the assumed or required representation of the data, while easy for humans to comprehend, is very difficult to implement in a computer system. The concepts behind the systems are sound, but implementations can only be application and data representation specific.

Much of the difficulty in creating an implementation for these systems lies in the vast number of calculations required to obtain enough data to reason on. Advances in hardware (such as faster CPUs and general purpose GPU computing) have opened the door to an era of unparalleled calculation potential. All hardware has limitations and at the very least, must be constrained by the speed of light. As such, relying on hardware to provide the needed performance is folly; the algorithms used to analyze the data must be as efficient as possible.

This collection of work attacks the problem from three key directions:

1. Improvement and optimization of the used algorithms.

2. Refinements to the QSR system (VRCC-3D+) itself.

3. Implementation and prototype application of the system.

This dissertation is formatted as follows. First, a review of current literature that concerns QSR systems is presented, followed by five papers accepted for publication or in review. In the first paper, a mechanism to allow dynamic error tolerance based on the finite nature of the pixel and the distance of the object from the observer in VRCC-3D+ is introduced. A method to dynamically build decision trees using a composition table to speed up the computation of the relations is presented in the second paper. The third paper is an exploration of one of many possible applications

of an implementation of VRCC-3D+. In the fourth paper, a dual graph representation of three dimensional objects is used to improve the quality and efficiency of bounding volume hierarchies. Finally, a refinement of the relations in VRCC-3D+ inspired by computational challenges is presented in the fifth paper.

2. REVIEW OF LITERATURE

2.1. REGIONS AND CONNECTION

2.1.1. RCC-8. Introduced in 1992 by Randall, Cohn, and Cui, RCC was initially an acronym for the names of the authors; later Region Connection Calculus was deemed a more appropriate name for what was being modeled. RCC-8 was the earliest system to define relationships between regions based on connection [52]. Randall, Cohn, and Cui define eight relationships (see Figure 2.1):

- Disconnected (DC)
- Externally Connected (EC)
- Partial Overlap (PO)
- Equal (EQ)
- Tangential Proper Part (TPP)
- Tangential Proper Part Inverse (TPPi)
- Non-Tangential Proper Part (NTPP)
- Non-Tangential Proper Part Inverse (NTPPi)

These relationships are Jointly Exhaustive Pairwise Disjoint (JEPD), meaning that there is no configuration of physically feasible regions that cannot be described by one of the relations, and no pair of regions can be described by more than one relationship (they are not ambiguous).

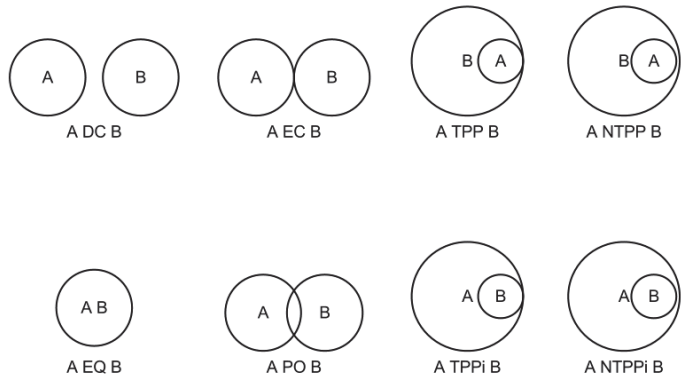


Figure 2.1: The RCC-8 Relationships (2D).

2.1.2. RCC-5. In 1994, Bennett simplified RCC-8 and encoded the relationships as propositional logic [11]. Bennett's initial premise was that while topological information and first order predicate calculi are more expressive, they are more difficult to use in automated reasoning. Bennett reduced the number of relationships in the original RCC from eight to five by removing distinction based on touching boundaries. The five relationships in RCC-5 are DiscRete (DR), PO, EQ, PP (Proper Part), and PPi (Proper Part inverse) (see Figure 2.2). He justifies the lessened expressive power of the system by stating that excessive expressive power wasn't necessary for some applications; if the problem of interest didn't require knowledge of tangential relationships, it didn't need to use relationships like EC or TPP/TPPi. Encoding the relationships using propositional logic increases the computational effectiveness of the system while lessening the expressive power overall. This weakening of the expressive power is due to the inability of the system to categorize the transition between DR and PO (for example). It is known that at some point the regions were externally connected, but that state was not captured by the system.

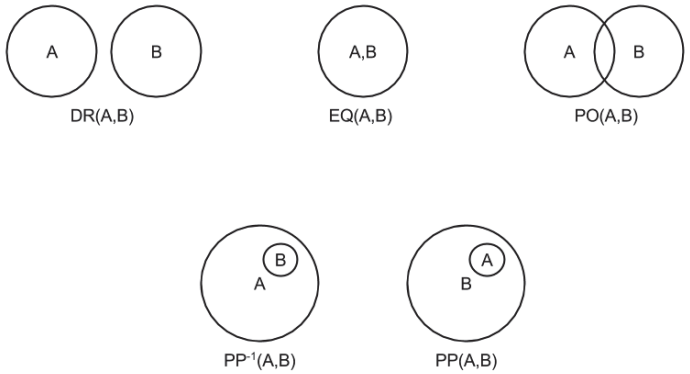


Figure 2.2: The RCC-5 Relationships (2D). Note the lack of distinction between Tangential and Non-Tangential Proper Part.

RCC-5 is a good system to use if only parthood or lack thereof is important in the problem set. RCC-5 is not capable of representing the moment at which a transition from DC to PO occurs. You can only determine when the regions are disconnected and partially overlapping, not when the change between those two states occurs. RCC-8 encodes that change as an additional state: EC.

2.1.3. RCC-23 and RCC-62. Two later developed systems, RCC-23 [17] and RCC-62 [48], took the opposite approach: they greatly increased the expressiveness of the system at the cost of automated reasoning ability. RCC-23 extends RCC-8 to have 23 relations that allow the system to handle concave regions differently than convex regions. The relationships themselves are based on the convex hull of the region (i.e., the minimal convex shape that encompases an arbitrary region). It then determines the relationships of the regions by whether they are Outside (O), Partially Inside (P) or Completely Inside (I) each other's concave region: a relationship in RCC-23 may look like $RCC23(A, B) = OOE$, meaning A is Outside of B's concave region, B is outside A's concave region, and they are externally connected.

The RCC-8 relationships are expanded using these convexity predicates: DC in RCC-8 encompasses eight relationships in RCC-23 (OOD, OPD, OID, PID, PPD, POD, IOD, IPD), while EC expands to nine RCC-23 relationships (OOE, OPE, OIE, PIE, PPE, POE, IOE, IPE, IIE) (see Figure 2.3).

RCC-62 handles concave shapes in a different manner: it breaks up regions into their boundaries, interiors, exteriors, and insides (for concave objects) (see Figure 2.4). The 62 relationships are defined by the intersections of these areas of interest. RCC-62 is more expressive than RCC-23 at the expense of ease of reasoning and computation.

RCC-23 and -62 excel when concavity becomes an issue. One example for which these systems would give more information about is shown in Figure 2.5. RCC-8 would not be able to differentiate these two configurations, while the more complex RCC systems would (i.e., RCC-8 would classify both configurations as DC(A,B), whereas RCC-23 would classify them as OIE and OOE, respectively, and RCC-62 would classify them as relationship 23 and 15, respectively, (see [48] for an explanation of these relationships).

2.1.4. RCC++. In 2007, Tiansi Dong exposed certain weaknesses in the RCC theories [19]:

1. It is not intended for all regions (for example, curves or regions extended with curves), it does not account for distance between regions.

2. It only allows two regions to be equivalent if the regions coincide in space (which follows from the distance relations argument).

To address these weaknesses, the author introduces the concept of categories of regions: regions of identical size and shape independent of the location. The author also presents several axioms, such that if two regions are connected, then there exists a region of every category that could connect the two regions. This axiom shows that a point can be a category of a region. The author claims that RCC++ is superior to

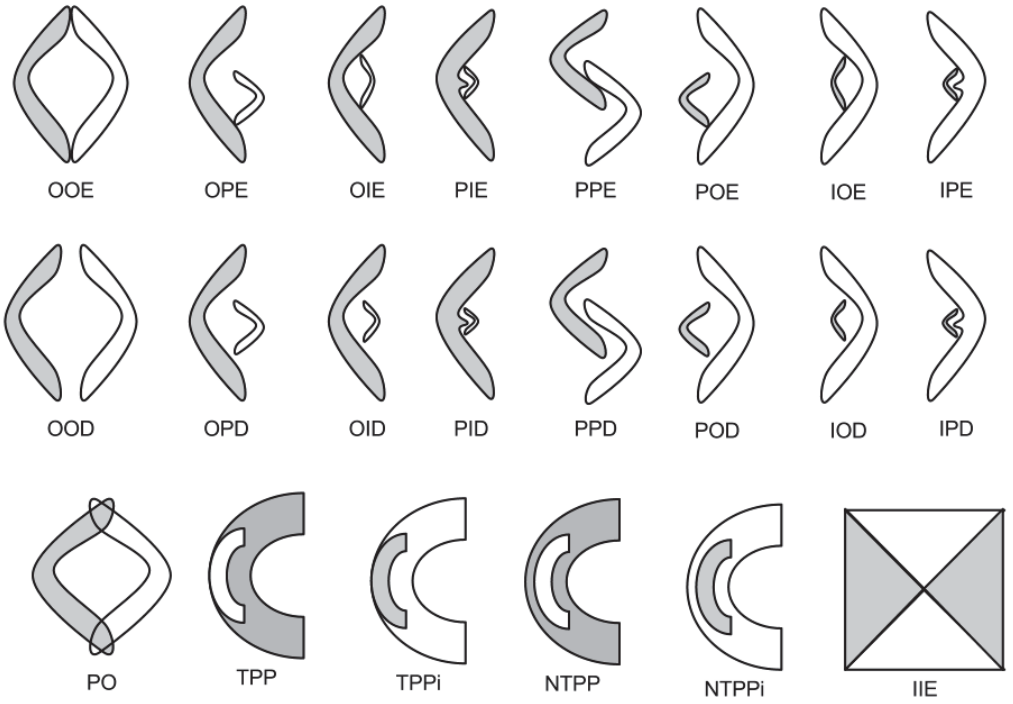
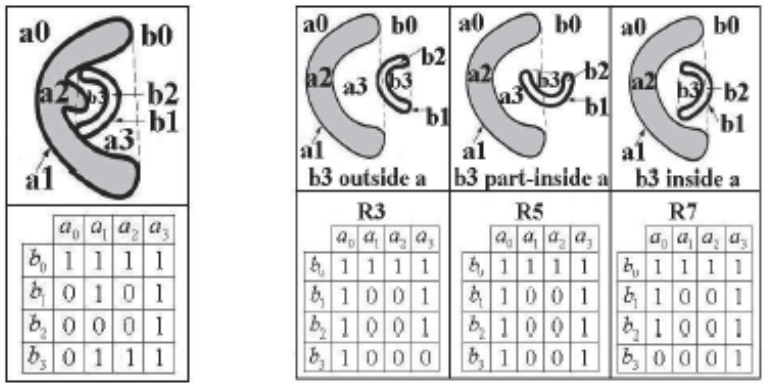


Figure 2.3: The 23 RCC-23 Relationships. Shaded regions are considered to be object A.



s

Figure 2.4: A Subset of the 62 RCC-62 Relationships [48]. a_0 is Object A's Outside, a_1 is Object A's Boundary, a_2 is Object A's Interior, and a_3 is Object A's Inside. Object B Has the Same Regions.

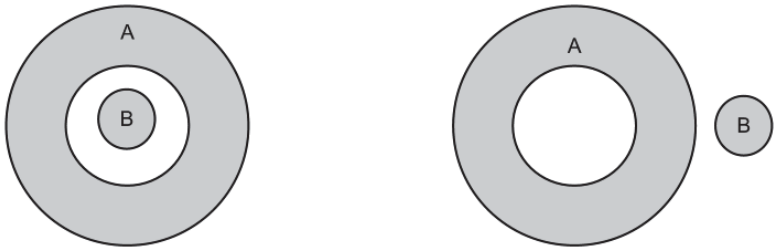


Figure 2.5: Two Systems RCC-8 Cannot Uniquely Classify.

the original RCC theory for all of the intended models in that it regards points as the natural atomic unit instead of the region. This may be true for some applications. However, as one of the weaknesses of the original theories was the inability to handle regions with zero area (see Figure 2.6), the improvements are purely theoretical, and do not translate to analyzing real world data, where zero volume regions cannot exist. Computationally, it remains similar to RCC-8, but the difficulty of reasoning using this theory increases.

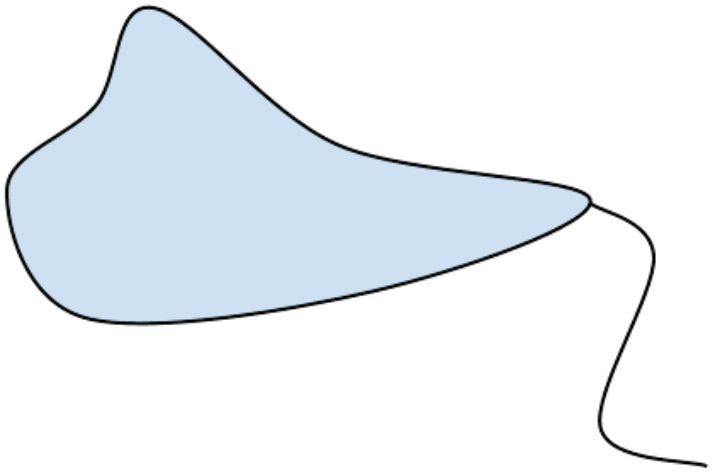


Figure 2.6: A System RCC++ Can Analyze, but RCC-8 Cannot.

2.2. OBSCURATION AND LINES OF SIGHT

2.2.1. LOS-14. LOS-14 [27] is a system introduced by A.P. Galton in 1994. It classifies regions based on what can be seen in the Lines Of Sight (LOS) from a given perspective. Fourteen relationships are defined based on obscuration (or the lack thereof) from a given viewpoint (see Figure 2.7).

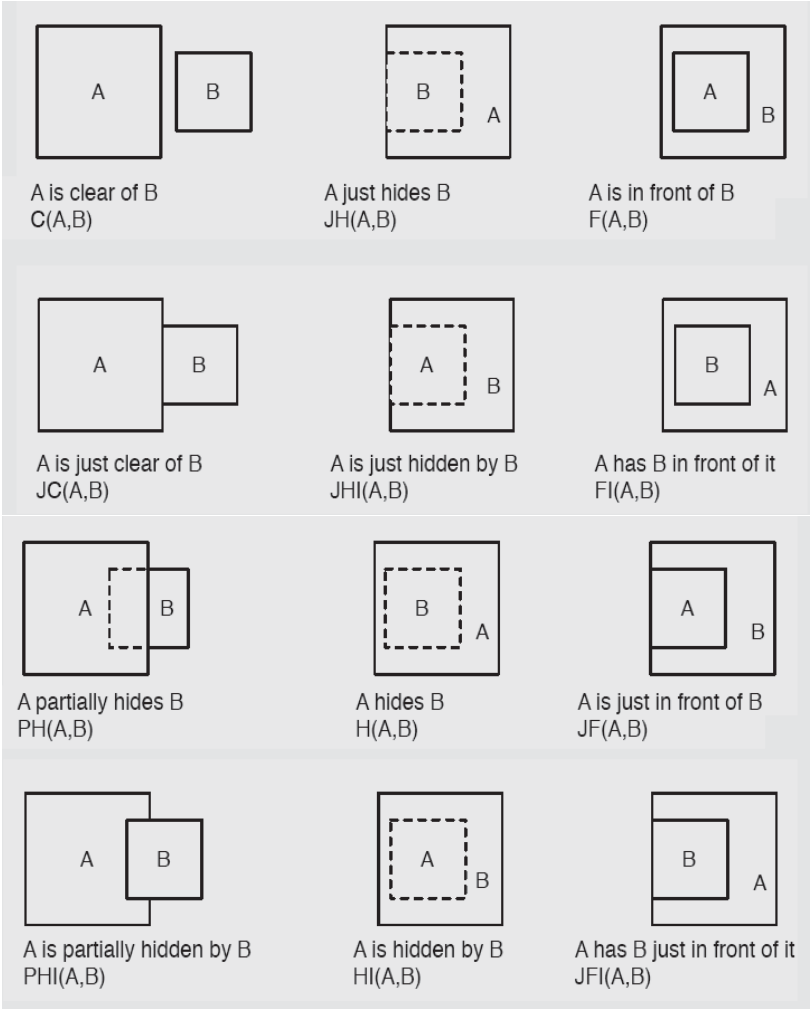


Figure 2.7: 12 of the LOS-14 Relations [27]. Omitted Relations are A Exactly Hides B ($EH(A, B)$) and A is Exactly Hidden by B ($IEH(A, B)$).

2.2.2. ROC-20. Another occlusion based calculus is ROC-20 [51]. It is similar to LOS-14, but extends it to add support for concave objects, which allows for mutual obscuration. Every spatial relationship in ROC-20 is defined in terms of the occlusion present and an RCC-8 relationship (see Figure 2.8). This system is significantly more expressive than LOS-14, and can apply to a greater number of cases, as it handles concave regions correctly.

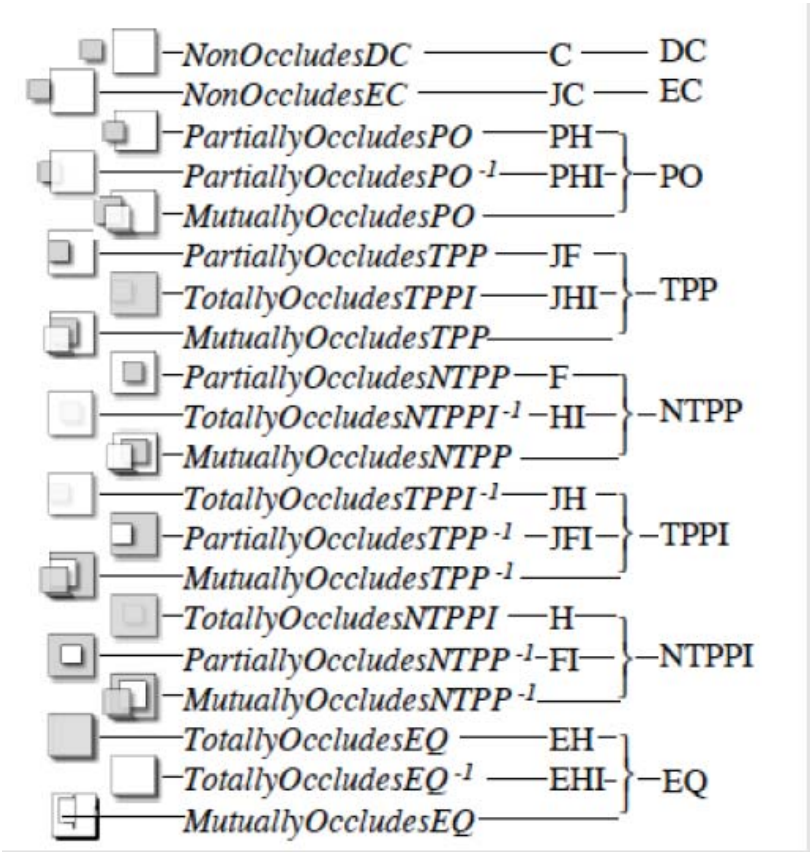


Figure 2.8: A Comparison of ROC-20, LOS-14, and RCC-8 [51]. In Each Example, Object A is Gray.

2.2.3. OCC. The Occlusion Calculus (OCC) was introduced by Kohler in 2002 [36] and characterizes relationships between objects by their respective projections into an image plane. The author states that the information obtained is only

from one perspective, and as such, this system should be paired with other QSR methods to get a fuller picture. The system sacrifices expressiveness for reduced computational and reasoning complexity.

2.2.4. OCS-14. Guha et al. introduced OCS-14 in 2011 [30]. OCS-14 was designed to correct for insufficiencies in earlier occlusion methods that made them infeasible for use in computer vision. Earlier methods had not accounted for whether the occluder was a moving object or part of the static background, and whether or not the visible part of an object was a connected blob or a fragment. As the OCS-14 is designed for computer vision, feasibility of computation is a concern, but not expressive power.

2.2.5. Interpreting Digital Images. Randal and Witkowski presented a work in 2006 [53] that explored the use of Occlusion methods such as OCC (among others) to analyze digital images. By defining aggregated pixels as regions, Region Collision Detectors can be used to determine the occlusion of the regions, allowing reasoning on the contents of the image. They provide algorithms for these collision detectors, but in doing so expose one of the biggest weaknesses in the methods presented thus far in this paper: they are generally restricted to working in two dimensions, while we live in a three dimensional world.

2.3. COMBINING METHODS FOR QSR

No single system can account for all aspects of the world as we see it. However, as stated in [36], sometimes systems can be combined to provide a more complete picture of the problem of interest, and cover the shortcomings of each system. In 2010, Albath et al. presented work on one such system.

2.3.1. RCC-3D. RCC-3D [8] is a system designed to consider three dimensions, be computationally feasible, and give the most comprehensive spatial information about the system possible. Initially designed for use in analyzing the evolution

of skeletal structures and other physical attributes, it also accommodates temporal reasoning. Sample implementations of the logic behind the spatial intersection predicates show the computational feasibility of the system, though large problem sets still resulted in runtimes upward of eight hours.

Because RCC-3D was to be used in visualizing physical changes over time, a GUI was deemed necessary. The resulting implementation was named VRCC-3D [7]. In implementation, certain ambiguities became apparent due to the finite nature of both object representation and computational precision. In one example, the program determined that the wings of an airplane were disconnected from the fuselage, when visually, they appeared to be externally connected. As such, some aspects of VRCC-3D needed to be redesigned.

2.3.2. VRCC-3D+. The result of this redesign is VRCC-3D+ [65]. As an implementation, VRCC-3D+ focuses on computational feasibility and automated reasoning power, but aims to sacrifice as little expressive power as possible. VRCC-3D+ uses compound relationships to describe the spatial interaction between objects. Each relationship consists of a 3D intersection relationship (one of the eight relationships from RCC-8, though computed in three dimensions) and a 2D obscuration term. In order to make the RCC-8 relationships valid in three space, the authors used a formulation of the definitions similar to those in [21], using the intersection of the boundaries, interiors, and exteriors to determine the appropriate 3D relationship. Similar information is gathered from the 2D projection onto an image plane for the obscuration term in the compound relationship. Tables 2.1 and 2.2 show the set theory formulations of the 3D and 2D relationships, respectively.

Table 2.1: Definition of 3D Intersections Through Intersection Predicates.

	IntInt	IntBnd	IntExt	BndInt	BndBnd	BndExt	ExtInt	ExtBnd
DC	False	False	True	False	False	True	True	True
EC	False	False	True	False	True	True	True	True
EQ	True	False	False	False	True	False	False	False
NTPP	True	False	False	True	False	False	True	True
NTPPc	True	True	True	False	False	True	False	False
PO	True	True	True	True	True	True	True	True
TPP	True	False	False	True	True	False	True	False
TPPc	True	True	True	False	True	True	False	True

Table 2.2: Definition of 2D Obscurations Through Intersection Predicates.

	IntInt	**IntExt**	**ExtInt**	o	o_c
pObs	T	T F	T	T	F
pObs_c	T	T	T F	F	T
pObs_e	T	T	T	F	F
pObs_m	T	T F T	F T T	T	T
eObs	T	F	F	T	F
eObs_c	T	F	F	F	T
eObs_e	T	F	F	F	F
eObs_m	T	F	F	T	T
cObs	T	T	F	T	F
cObs_c	T	F	T	F	T
cObs_e	T	T F	F T	F	F

The reason there are not 72 possible relations in VRCC-3D+ is that not every obscuration term is possible for every RCC-8 intersection term. For example, if two objects are disconnected, they cannot be equal distance from the camera and have the projections exactly overlap (eObs_e), as that would make the objects sharing the same space exactly (EQ). The 37 possible relationships are shown in Table 2.3.

Table 2.3: The Possible Obscurations for RCC-8 Relationships.

	nObs	nObs_c	nObs_e	pObs	pObs_c	pObs_e	eObs	eObs_c	eObs_e	cObs	cObs_c	cObs_e
DC	*	*	*	*	*		*	*		*	*	
EC	*	*	*	*	*		*	*		*	*	
PO				*	*	*	*	*		*	*	*
EQ									*			
TPP								*	*		*	*
TPPc							*		*	*		*
NTPP											*	
NTPPc										*		

Significant work [63, 24, 43, 23] continues to make VRCC-3D+ a computationally feasible and correct system through the use of composition tables, dynamic error tolerances, and more intelligent intersection algorithms to reduce the number of computations required to determine the relationships between an arbitrary number of objects in a scene. With 37 relationships, VRCC-3D+ is not as simple as RCC-8 in automated reasoning, but efforts continue to improve the model and its implementation in that respect [43, 25]. The computational efficiency of the algorithms used in the implementation have been examined and improved up through more intelligent primitive partitioning in the bounding hierarchy creation [26].

PAPER

I. Efficient Computation of Object Boundary Intersection and Error Tolerance in VRCC-3D+

Nathan Eloe *Jennifer Leopold* *Chaman Sabharwal*

nwe5g8@mst.edu *leopoldj@mst.edu* *chaman@mst.edu*

Zhaozheng Yin

yinz@mst.edu

Missouri University of Science and Technology

ABSTRACT

Computational geometry is a field that is relevant to computer graphics rendering, computational physical simulation, and countless other problem domains involving the use of image data. Efficiently determining the intersection of the boundaries, interiors, and exteriors of two objects can mean the difference between a realistic and relevant simulation, and a slow program that produces results that do not keep pace with user manipulation of the object. However, the speed of these calculations is not the only area of concern. Taking into consideration the finite unit of resolution in a computer display (the pixel) and error in the floating-point representation of numbers, it may be the case that the perceived correctness of these computations does not necessarily correspond to the accuracy with which the calculations are carried

out. In this paper, we examine two of the most well-known methods of determining such intersections, as well as various programming language libraries available to perform these calculations. These existing approaches are considered with respect to limitations in human perception, display resolution, and floating point error. We also propose a new method which lends itself to exploiting the inherently parallel nature of these calculations.

1. INTRODUCTION

VRCC-3D+ [11, 12, 6, 7, 5] is a mathematical model that describes relations between three dimensional objects in space in terms of the connectivity and obscuration between each pair of objects in a scene. The system facilitates knowledge discovery by determining possible intermediate configurations of the objects from one state to another. Implementing the VRCC-3D+ mathematical model resulted in computational inefficiency (unacceptable delays between initial model loading and the calculation of the spatial relations between all pairs of objects). Our investigation of a more efficient method to perform these computations focused on determination of the intersections between the interiors, exteriors, and boundaries of the objects.

2. VRCC-3D+INTERSECTIONS

The VRCC-3D+ system distinguishes eight types of 3D connectivity relations using eight intersections that involve the boundary, interior, and/or exterior of one object with those of another (see Table 2.1).

Connectivity in 3D is only one part of each (composite) VRCC-3D+ relation. The other part of each VRCC-3D+ relation characterizes the obscuration between the two objects of interest, which can be none, partial, or complete (as well as converse

Table 2.1: Definition of 3D Spatial Relationships Using Intersection Predicates. Φ denotes empty intersection set, $\neg\Phi$ denotes non-empty intersection set.

	IntInt	IntBnd	IntExt	BndInt	BndBnd	BndExt	ExtInt	ExtBnd
DC	Φ	Φ	$\neg\Phi$	Φ	Φ	$\neg\Phi$	$\neg\Phi$	$\neg\Phi$
EC	Φ	Φ	$\neg\Phi$	Φ	$\neg\Phi$	$\neg\Phi$	$\neg\Phi$	$\neg\Phi$
EQ	$\neg\Phi$	Φ	Φ	Φ	$\neg\Phi$	Φ	Φ	Φ
NTPP	$\neg\Phi$	Φ	Φ	$\neg\Phi$	Φ	Φ	$\neg\Phi$	$\neg\Phi$
NTPPc	$\neg\Phi$	$\neg\Phi$	$\neg\Phi$	Φ	Φ	$\neg\Phi$	Φ	Φ
PO	$\neg\Phi$	$\neg\Phi$	$\neg\Phi$	$\neg\Phi$	$\neg\Phi$	$\neg\Phi$	$\neg\Phi$	$\neg\Phi$
TPP	$\neg\Phi$	Φ	Φ	$\neg\Phi$	$\neg\Phi$	Φ	$\neg\Phi$	$\neg\Phi$
TPPc	$\neg\Phi$	$\neg\Phi$	$\neg\Phi$	Φ	$\neg\Phi$	$\neg\Phi$	Φ	Φ

relations for each). Obscuration in VRCC-3D+ is determined by the overlap of the projections into the image plane and the object depth. The techniques (Triangle Triangle intersection, AABB Bounding Boxes) and considerations (error tolerance and speed) discussed in this paper will apply to the calculations of both the connectivity and obscuration parts of the VRCC-3D+ relations.

Definition of the spatial relations using Table 2.1 results in an over-determined system [11]. A symmetry argument also applies. The intersection of the interior of object A and the boundary of object B is the same as the intersection of the boundary of object B and the interior of object A: IntBnd(A,B)=BndInt(B,A). Thus, an efficient algorithm for IntBnd will provide an efficient algorithm for BndInt, and vice versa. This also holds for the intersections of boundaries and exteriors.

These four intersection predicates are insufficient to uniquely distinguish all relationships. For example, DC and EC are indistinguishable using these four predicates, and require either the BndBnd or the IntInt predicate. Computing the intersection of the boundaries allows us to determine the 3D relationships of two objects while avoiding unnecessary computation to determine the interior or exterior. The remaining predicates that do not involve the boundaries are unnecessary and inefficient for our implementation.

3. COMPUTATION OF BOUNDARY INTERSECTION

A naive method of computing the boundary intersection predicate is to use an algorithm such as that presented in [9] to determine pairwise intersection between the triangular faces of objects A and B: an $O(f_a f_b)$ algorithm with f_a and f_b as the number of faces in objects A and B. A more sophisticated method, AABB trees, uses nested axis-aligned bounding boxes to decrease the number of faces that require the intersection calculation. The use of trees presents an opportunity to reduce the complexity by reducing the number of calculations. CGAL [4] is a computational geometry toolkit that uses AABB trees to implement intersection and distance algorithms. Table 3.1 shows the runtimes of the triangle intersection and the CGAL methods on two spheres (see Figure 3.1), each of which has approximately 2000 faces. All timing was done on an AMD Bulldozer processor running at 3.1Ghz. Results were obtained using a C module and the CGAL bindings for Python and averaged 100 runs.

Table 3.1: Intersection Test Runtimes.

Implementation	Average Runtime (s)
C Triangle Intersection	12.0349209094
CGAL AABB	0.186076021194

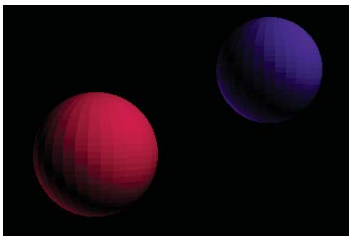


Figure 3.1: Two Disconnected Objects (DC).

4. HUMAN PERCEPTION AND FLOATING POINT ERROR

There is no ambiguity in Figure 3.1 regarding the intersection of boundaries. However, in a more complicated case such as that shown in Figure 4.1, more rigorous calculations are required. Upon cursory examination of the rendering of the airplane, it appears that all of the wings are attached to the fuselage. However, according to the computational intersection of boundaries, the wings and the fuselage do not intersect: they cannot be considered externally connected.

While the calculations are being carried out accurately, they do not reflect the cognitive perception of the image. This could be attributed to several factors such as errors in the representation of floating point numbers in both the viewer display and model generation software, or the minimum distance that can be represented in rendering to a monitor with finite resolution.

To remedy this, we must allow some tolerance: the shortest distance between the two planes must be less than some small value, ϵ. Implementing this in the direct triangle intersection test is trivial. However, using the CGAL AABB implementation requires a significant change in the algorithm itself: instead of determining whether

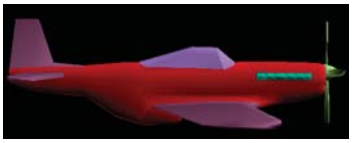


Figure 4.1: Model of an Airplane.

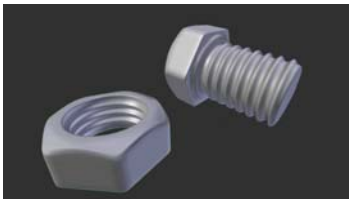


Figure 4.2: Nut and Bolt Model.

the triangles intersect, we need to find the minimum distance between the two closest faces. One way to do this is to determine all of the segments that make up the faces of the object and find the closest distance to a face in the other object.

Table 4.1 shows the timings for these modified epsilon tolerance tests. Both algorithms also were used to determine the intersection of a high resolution depiction of a nut and bolt (Figure 4.2).

Table 4.1: Intersection Runtimes Using ϵ Tolerance.

	Airplane	Spheres	Nut and Bolt
Obj1 Name / Face Count	Wings_4 / 18	A / 1984	Helix01 / 43872
Obj2 Name / Face Count	Prop_4 / 144	B / 1984	Helix02 / 27432
C Triangle Intersection Runtime(s)	0.015720	12.017024	388.967400
CGAL AABB Runtime (s)	0.0110691	0.23603410	86.336132

5. ϵ DETERMINATION

Once a tolerance test was developed for the intersection, it was necessary to determine an appropriate value of epsilon. Trial and error yielded an ϵ of approximately 10^{-4}. Appropriate values of epsilon will change based on the precision with which the values are stored in the model and the finite nature of the pixels that comprise the image. Of these two factors, we can both calculate and control the effect of the latter at the viewer level. Knowing the distance in world units from the virtual camera to the viewing plane (f in world units), the horizontal width in pixels of the scene (h in world units, w in pixel count), and the field of view angle (θ) allows the calculation of the physical size of the pixel at the image plane (see Figure 5.1). We can determine this value as follows. Let "px" be pixel units and "units" be world units:

$$h = f * tan\left(\frac{\theta}{2}\right)$$

We calculate the size in world units per pixel as

$$\frac{h}{\frac{w}{2}}\frac{units}{px} = \frac{2h}{w}\frac{units}{px}$$

The size of one pixel on the image plane is then

$$\epsilon_f = \frac{2h}{w}\frac{units}{px} * 1px = \frac{2h}{w}units = \frac{2f}{w}tan\left(\frac{\theta}{2}\right)units$$

The default visualizer parameters yield a value of approximately $1.4 * 10^{-3}$, which is a greater tolerance than the $1 * 10^{-4}$ that was found through manual experimentation. This value of ϵ gives the perceived results for Figure 4.1, and is only dependent upon the position of the objects and the resolution of the viewer window.

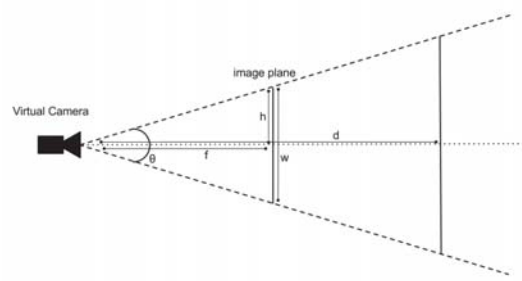


Figure 5.1: Camera and Image Plane.

The minimum visible distance is directly proportional to the object distance from the camera. This calculation can be broken into two cases: the two objects have equal depth from the image plane, and the two objects have differing depths. The first case is trivial: given two objects of depth d from the camera, we use similar triangles (as depicted in Figure 5.1) to obtain the smallest distinguishable distance, ϵ_w, at that depth:

$$\frac{\epsilon_f}{f} = \frac{\epsilon_d}{d}$$

Since

$$\begin{aligned} \frac{\epsilon_f}{f} &= \frac{\frac{2f}{w}tan(\frac{\theta}{2})}{f} \\ &= \frac{2}{w}tan(\frac{\theta}{2}) \end{aligned}$$

we can calculate the epsilon value at a depth d from the camera as

$$\epsilon_d = \frac{2d}{w}tan(\frac{\theta}{2})$$

This result can be reinterpreted to generalize it to all points regardless of distance to the camera. The above states that due to the finite nature of the pixel, we can define a sphere around any point with radius $\frac{\epsilon_w}{2}$ in which we cannot determine the actual position of the point (see Figure 5.2). This concept of a *probability cloud* is frequently used in Quantum Physics to approximately describe the locations of electrons in given states. We say that two points $P1$ and $P2$ are indistinguishable if their probability clouds overlap; that is, if $|P1 - P2|$ is less than or equal to the sum of the radii of their probability clouds. If ϵ_1 and ϵ_2 are the respective epsilon values computed for these points $P1$ and $P2$, the resulting tolerance is:

$$\epsilon = \frac{\epsilon_1}{2} + \frac{\epsilon_2}{2} = \frac{\epsilon_1 + \epsilon_2}{2}$$

In Figure 5.2, the points $C1$ and $C2$ are considered to be within the error tolerance because the probability clouds overlap. The same holds for points $C2$ and $C3$. $C1$ and $C3$ are not within the error tolerance even though $C3$ is at the same depth as $C2$.

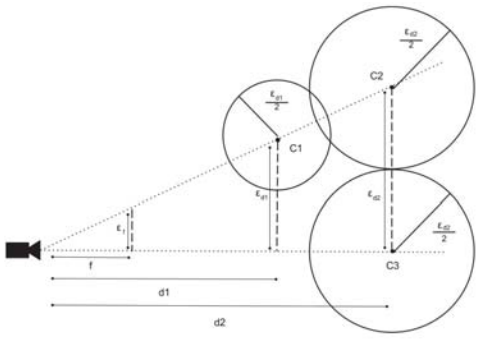


Figure 5.2: Using a Probability Cloud to Dynamically Determine ϵ Tolerance.

6. FUTURE WORK

This problem space is inherently parallel: the intersection of any pair of faces is independent of the intersection of any other pair of faces. The large number of similar calculations on a large, but static, set of data should be enough to overcome the primary downfall of using general purpose GPU computing technologies (e.g., OpenCL [2], CUDA [3] by NVIDIA, or Stream [1]) by AMD: the time it takes to transfer the data to the graphics card [8]. For the intersection of triangles method to be as fast as AABB trees, we require a speedup of 50X (e.g. $\frac{12.02}{0.236}$ for the spheres), which is well within the plausible speedups reported in [10].

A heavily parallelized AABB algorithm may also work, but would not necessarily be as cost efficient due to the recursive nature of trees. Exploration of an OpenCL implementation of these algorithms may even allow these computations to be done efficiently on mobile devices with sufficiently new integrated graphics.

7. SUMMARY

Being able to efficiently calculate the intersections between the boundaries, interiors, and exteriors of 3D objects introduces new ways to program simulations, collisions, and model transformations over time. Exploiting the ability of VRCC-3D+ to identify impossible states and eliminating some calculations would lead to more efficient collision detection in game and simulation engines, and modeling software.

In this paper we have explored two methods of calculating these intersections and discussed methods of dynamically determining the error tolerance. These strategies will help to make spatial reasoning applications such as VRCC-3D+ more practical for knowledge validation and discovery in real-time.

References

[1] ATI Stream Technology. http://www.amd.com/stream, visited 2012-04-27.

[2] Chronos Group: OpenCL. http://www.khronos.org/opencl/, visited 2012-04-27.

[3] NVIDIA Technologies: CUDA. http://developer.nvidia.com/category/zone/cuda-zone, visited 2012-04-27.

[4] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org, visited 2014-10-27.

[5] Julia Albath, Jennifer Leopold, Chaman Sabharwal, and Kenneth Perry. Efficient Reasoning with RCC-3D. In *The 4th International Conference on Knowledge Science, Engineering, and Management (KSEM 2010)*, pages 470–481. KSEM, September 2010.

[6] Julia Albath, Jennifer L. Leopold, and Chaman L. Sabharwal. Visualization of Spatio-Temporal Reasoning Over 3D Images. In *Proceedings of the 2010 International Workshop on Visual Languages and Computing (in conjunction with the 16th International Conference on Distributed Multimedia Systems)*, DMS '10, pages 277–282, 2010.

[7] Julia Albath, Jennifer L. Leopold, Chaman L. Sabharwal, and Anne M. Maglia. RCC-3D: Qualitative Spatial Reasoning in 3D. In *Proceedings of the 23rd International Conference on Computer Applications in Industry and Engineering*, pages 74–79, November 2010.

[8] David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2010.

[9] Tomas Möller. A Fast Triangle-Triangle Intersection Test. *journal of graphics, gpu, and game tools*, 2(2):25–30, 1997.

[10] Antonio J. Rueda Ruiz and Lidia M. Ortega. Geometric Algorithms on CUDA. In *GRAPP*, pages 107–112, 2008.

[11] Chaman Sabharwal and Jennifer L. Leopold. Reducing 9-Intersection to 4-Intersection for Identifying Relations in Region Connection Calculus. In *The 24th International Conference on Computer Applications in Industry and Engineering*, pages 118–123. CAINE, November 2011.

[12] Chaman L. Sabharwal, Jennifer L. Leopold, and Nathan Eloe. A More Expressive 3D Region Connection Calculus. In *Proceedings of the 2011 International Workshop on Visual Languages and Computing (in conjunction with the 17th International Conference an Distributed Multimedia Sysems (DMS 11))*, pages 307–311, August 2011.

II. Efficient Determination of Spatial Relations Using Composition Tables and Decision Trees

Nathan Eloe *Jennifer Leopold* *Chaman Sabharwal*

nwe5g8@mst.edu *leopoldj@mst.edu* *chaman@mst.edu*

Douglas McGeehan

djmvfb@mst.edu

Missouri University of Science and Technology

ABSTRACT

In order for Qualitative Spatial Reasoning applications to be both useful and usable, the information feedback loop between the computational engine and the user must be as seamless as possible. Inherently, computational geometry can be quite expensive, and every effort must be made to avoid inefficient or unnecessary calculations. Within the field of Region Connection Calculi, the 9-Intersection model often is used to determine the spatial relation between two regions. Consequently, optimization efforts typically focus on calculations involving the intersections between the interiors, boundaries, and exteriors of the regions, or the use of composition tables to narrow down the possibilities for the relations that can hold between two regions. The few implementations of spatial reasoners that have been attempted have been

simply proofs-of-concept and/or have been limited to two dimensions. Herein we present a novel approach that combines the use of composition tables and decision trees to efficiently determine the spatial relation between two objects in 3D considering both connectivity and obscuration. This approach has been fully implemented for the VRCC-3D+ spatial reasoning system, and benchmarks are included to corroborate our claims of efficiency.

1. INTRODUCTION

Qualitative Spatial Reasoning (QSR) has the potential to further enhance the functionality of applications for diverse fields such as Geographic Information Systems (GIS), visual programming language semantics, and digital image analysis. Unfortunately, the practicality of automated spatial reasoning may be diminished if expensive and unnecessary calculations are not avoided; every delay in the information feedback loop inhibits not only real-time user interaction, but also knowledge discovery.

The foundation of many QSR systems are Region Connection Calculi (RCC), which use a 9-Intersection model to distinguish spatial relations based on the intersections of the interiors, boundaries, and exteriors of two or more regions. It has been shown that the 9-Intersection model can be reduced to a 4-Intersection problem [13] through the use of a decision tree. Although examining fewer intersections provides a tremendous savings in computational effort, the amount of time to determine the spatial relation between two regions still can be unacceptable in real time if additional optimizations are not employed.

There are a variety of different RCC models, each with different degrees of expressivity. Unfortunately, few actual implementations of RCC systems exist. Those that do exist are incomplete (e.g., Albath's original RCC-3D system [2]), still in the alpha or proof-of-concept stage [17, 15], or limited to two dimensions [17]. Many

libraries in existence that allow computation of the 9-Intersection model, and thus computation of the RCC-8 relations, also share these limitations [7] or require special representations of data [16, 6, 18]. Not only has the lack of full implementations limited the actual use of these models, it also has concealed many of their computational challenges.

Herein we present a novel approach for efficiently calculating spatial relations. It utilizes decision trees and composition tables to avoid as many unnecessary computations as possible. This approach has been implemented for a QSR system (VRCC-3D+ [15]) that determines the spatial relation between two objects in 3D considering both connectivity and obscuration. We also provide benchmarks to show the viability of our approach.

2. BACKGROUND AND RELATED WORK

2.1. RCC-8

Introduced in 1992 by Randall, Cohn, and Cui, RCC was initially an acronym for the names of the authors; later Region Connection Calculus was deemed a more appropriate name for what was being modeled. RCC-8 was the earliest system to define relationships between regions based on connection [11]. Randall, Cohn, and Cui define eight relationships (see Figure 2.1):

- Disconnected (DC)
- Externally Connected (EC)
- Partial Overlap (PO)
- Equal (EQ)
- Tangential Proper Part (TPP)

- Tangential Proper Part Converse (TPPc)
- Non-Tangential Proper Part (NTPP)
- Non-Tangential Proper Part Converse (NTPPc)

These relationships are Jointly Exhaustive Pairwise Disjoint (JEPD), meaning that there is no configuration of physically feasible regions that cannot be described by one of the relations, and no pair of regions can be described by more than one relationship (they are not ambiguous).

2.2. VRCC-3D+

VRCC-3D+ [15] is a GUI for and an extension of the RCC-3D [2] system that was designed to maximize both computational feasibility and the comprehensiveness of resulting information. It defines relations in three dimensions using a composite relation of the form R_O(A, B) for two regions A and B. The R part of the VRCC-3D+ relation (herein referred to as the base relation) is one of the eight RCC-8 relations, although computed in three dimensions, not two dimensions.

The O part of the VRCC-3D+ relation represents obscuration as determined for a particular two dimensional projection plane; there are fifteen possible obscuration relations, characterized by intersections for the interiors and exteriors of two regions, and a qualitative depth parameter (see Table 2.1). An obscuration relation has a base type of either No Obscuration (nObs), Partial Obscuration (pObs), Complete Obscuration (cObs), or Equal Obscuration (eObs). Additional qualifiers, including converse obscuration (_c) or equal depth from camera (_e) are added to further enhance the expressive power of the obscuration relation. Every obscuration relation is conversely related to exactly one other obscuration. Not every obscuration

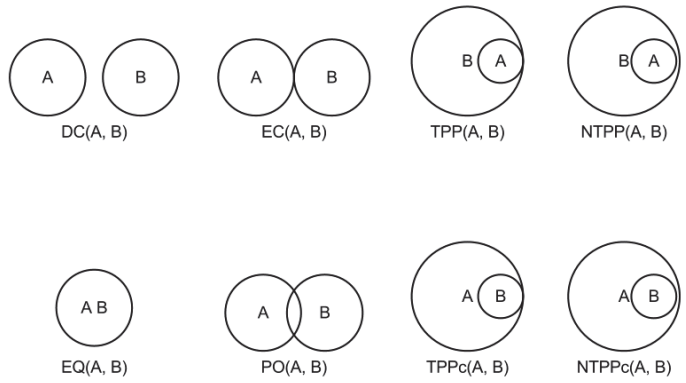


Figure 2.1: Examples of the Eight JEPD RCC-8 Relations.

relation is applicable to every base relation; in all, there are 46 VRCC-3D+ relations (as shown in Table 2.2). See [15] for a more detailed discussion of the VRCC-3D+ model.

2.3. DECISION TREES

The field of research in Top-Down Induction of Decision Trees produced many algorithms for the creation of such trees. Iterative Dichotomiser 3 (ID3) [9] creates a simple and minimal decision tree from a sample of classified training data. Traversing the tree leads to a leaf node that provides a classification for a given entity based on the values of certain attributes of the entity. The construction of the tree requires that each non-leaf node makes a decision based on the attribute that results in the greatest gain of information. ID3 produces a tree by using the informational entropy of the training set; this is the amount that making a decision decreases the entropy in the sub groups generated by splitting the training set based on the value of the attribute.

Table 2.1: Characterization Table for Obscuration Relations [14]. Int = interior, Ext = exterior; F = No (empty intersection), T = True (Non-empty) intersection. InFront = Y means A is closer to the viewing plane than B. N means B is closer than A. E means mutual obscuration or A and B are equidistant to the viewing plane.

	IntInt	IntExt	ExtInt	InFront
nObs	F	F	F	Y
nObs_c	F	F	F	N
nObs_e	F	F	F	E
pObs1	T	F	T	Y
pObs2	T	T	T	Y
pObs_c1	T	T	F	N
pObs_c2	T	T	T	N
pObs_e	T	T	T	E
eObs	T	F	F	Y
eObs_c	T	F	F	N
eObs_e	T	F	F	E
cObs	T	T	F	Y
cObs_c	T	F	T	N
cObs_e1	T	T	F	E
cObs_e2	T	F	T	E

Table 2.2: The Possible Obscurations for RCC-8 Relationships [14].

	DC	EC	PO	EQ	TPP	TPPc	NTPP	NTPPc
nObs	*	*						
nObs_c	*	*						
nObs_e	*	*						
pObs1	*	*	*					
pObs2	*	*	*					
pObs_c1	*	*	*					
pObs_c2	*	*	*					
pObs_e			*					
eObs	*	*	*			*		
eObs_c	*	*	*		*			
eObs_e			*	*	*			
cObs	*	*	*			*		*
cObs_c	*	*	*		*		*	
cObs_e1			*		*	*		
cObs_e2			*		*	*		

Decision trees have been used previously to classify spatial relations. Sabharwal et al. [13] used decision trees to reduce the 9-Intersection characterization of spatial relations to 4-Intersections. A decision tree for classifying a relation in RCC-8 is shown in Figure 2.2. This decision tree allows for identification of an RCC-8 relation in as few as two truth predicates, or at most four truth predicates. Clementini et al. [3] also proposed decision trees as the standard way of classifying relations, and gave multiple decision trees for the eight RCC-8 relations, based on different ways of calculating the probability of a specific intersection being empty.

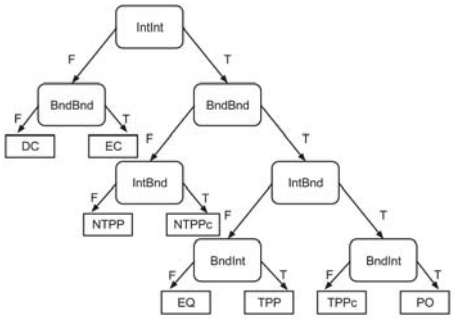


Figure 2.2: The Hand Generated Decision Tree [13] Hierarchy of the Intersection Predicates for the RCC-8 Relations.

While having a decision tree might increase the ease and efficiency of spatial relation determination, manual creation of the tree might be difficult and error prone, particularly for some of the RCC models that have many relations (e.g., RCC-23 [4] has 23 relations, VRCC-3D+ [1] has 46 relations, and RCC-62 [8] has 62 relations). A decision tree also exposes another problem: the predicates are not all equal in their execution time or complexity. For example, the first decision node in the tree proposed by Sabharwal et al. [13] is IntInt. This is because IntInt distinguishes DC and EC from all of the other RCC-8 relations (i.e. PO, EQ, TPP, NTPP, TPPc, and NTPPc). However, in the current implementation of VRCC-3D+, any intersection calculation

that does not involve the boundary (i.e., IntInt) is avoided. This decision is influenced by the fact that our datasets are Wavefront OBJ formatted files that only provide boundary information. Other factors such as whether a GPU (e.g. OpenCL, CUDA, or Stream) is being utilized could influence the decision to prioritize the computation of certain predicates over others in a particular implementation.

Our goal is to develop a method by which the nodes in the decision tree can be dynamically (rather than manually) generated based on the relations in the composition table. Recall that given objects (or regions) A, B, and C, and knowledge of relations R(A,B) and R(B,C), the composition table entry R(A,B) $\circ$ R(B,C) is the set of all possible relations for R(A,C). We want to avoid complete pre-computation of the decision trees for the following reasons:

- A general mechanism for deciding the most appropriate decision predicate can be adapted for use with other RCC systems (e.g., RCC-23 and RCC-62), as well as for relations that do not simply use binary terms in the predicates (e.g., in VRCC-3D+ the obscuration relations utilize a ternary term, InFront).

- Intersection predicates are not necessarily equal in terms of computational efficiency. A sequential execution core (standard CPU) may have an efficient algorithm for implementing the intersection of interiors. However, a massively parallel computation platform (GPGPU through CUDA, OpenCL, or Stream) may be able to perform a naïve pairwise facial intersection test (e.g., for BndBnd) much faster than it could perform a tree implementation for IntInt. A generalized predicate selection algorithm can more easily be modified to utilize whichever predicates execute most efficiently for a particular hardware configuration.

- There are 247 total subsets of the RCC-8 relations of cardinality 2 through 8. However, in the composition table, a relatively small percentage (≈10%) of these subsets actually occur. Precomputation of all 247 (or indeed even the actual subsets) would be an inefficient use of memory and effort.

It should be noted that although this discussion has focused on RCC-8 relations (which represent the base relations in VRCC-3D+), the method presented herein is not limited to RCC-8, and can be extended directly to any RCC system in which the relations can be represented as combinations of binary predicates.

3. IMPLEMENTATION

3.1. ALGORITHM

Quinlan's ID3 [9] generates decision trees based on a training set of data, classifying inputs based on attributes. Quinlan also introduced another decision tree algorithm, C4.5 [10], that addresses some of the weaknesses in ID3, including: (1) trees can become excessively large, depending on the number of attributes and classes in the system, and (2) some decision paths may result in unknown decisions if the training set is not complete.

We mitigate these problems by having a very small number of attributes (i.e., the intersection predicates), and having complete knowledge of the determination of the relations with respect to the predicates. As such, we can show that no matter what subset of RCC-8 relations we want to be able to differentiate between, we can come up with a tree of decision nodes of finite size that will always yield a single RCC-8 relation as a terminating node. The additional overhead of C4.5 over ID3, although negligible, is unnecessary for our application. Consequently, we use the attribute selection step of ID3 to allow selection of the most informative predicate to calculate.

Generating the entire decision tree for a subset of RCC-8 relations is inefficient: with unambiguous binary predicates, determining the order of the decision nodes for the unused sub-tree would be wasted computation. Only determining the decision nodes along the path to the bottom of the tree also would not result in a useful representation.

In the following pseudo-code, truePreds is a hash map, containing key-value pairs where the key is an intersection predicate and the value is the set of possible relations that result from the intersection predicate, BndInt, being true (e.g. truePreds[BndInt] = {NTPP, PO, TPP}).

```
predicates = {IntInt, IntBnd, BndInt,
              BndBnd, BndExt, ExtBnd}


def ratio(S, pred):
  intersection = S & truePreds[pred]
  return |intersection| / |S|


def entropy(S, pred):
  if |S| < 2:
    return 0.0
  sum = 0.0
  r = ratio(S,pred)
  //if r is 1 or 0, that means it does
  //not contribute to the entropy
  //and we have already
  //partitioned on that predicate
  if r > 0:
    sum += r * lg(r)
```

```
  if 1-r > 0:
    sum += (1-r) * lg(1-r)
  return -sum

def entropy(S):
  if |S| < 2:
    return 0.0
  return lg(|S|)

def gainRatio(S, pred):
  if |S| == 0:
    return 0.0
  branchT = S & truePreds[pred]
  branchF = S - truePreds[pred]
  tEntropy = entropy(branchT)
  fEntropy = entropy(branchF)
  splitH = |branchT| * tEntropy
  splitH += |branchF| * fEntropy
  splitH /= |S|
  if splitH > 0:
    return (entropy(S) / splitH) - 1
  return 0.0
```

We can then rank the predicates by their informational gain ratio. The predicate with the highest gain ratio is the predicate that should be chosen next to calculate.

This algorithm presents another route with which we can shorten the information feedback loop. Optimization of algorithms is still a necessity, but the potential gain from doing so is lost if time is wasted calculating uninteresting or redundant predicates. Again we emphasize that the overall goal is to determine the spatial relation between two objects in space, and to do so in the most efficient way possible. Previous efforts have addressed this problem from two different directions. Composition tables [11] reduce the number of possible relations by using global information about the relations of the objects relative to other objects in the scene. Decision trees, specifically hand generated decision trees, reduce the problem from 9-Intersection predicates to at most 4-Intersection predicates [13] but do not take global information into account. By algorithmically selecting the decision nodes of the decision tree as needed we aim to combine these methods and use both global information (by iteratively using the global information in the composition table) and local information (the use of decision nodes) to generate the relation between two objects.

By using a function (CTFilter(A,B)) that allows us to use global information to reduce the number of possible relations, we can determine the relationship between two objects now as follows:

```
def calcRCC8(A,B):
  possibleRels = CTFilter(A,B)
  while |possibleRels| > 1:
    //predicates sorted on decreasing
    //information gain ratio
    preds = sorted predicates
    p = calculate preds[0]
    possibleRels = {possibleRels | p}
  return possibleRels
```

This algorithm uses the composition table to use global information to reduce the possible number of relations between objects A and B. As long as the number of relationships in the resulting list is greater than one, the predicates are sorted based on decreasing information gain ratio, the first (most useful) predicate is calculated, and the relations are filtered such that possibleRels only contains the relations in which the calculated predicate holds. These filtered relations are used in the next iteration.

Table 3.1 shows the gain ratios obtained for five intersection predicates for three different subsets of RCC-8 relations, each subset obtained from the RCC-8 composition table: TPPc ° TPP = {PO, TPP, TPPc, EQ}, TPP ° TPPc = {DC, EC, PO, TPP, TPPc, EQ}, and TPPc ° EC = {EC, PO, TPPc, NTPPc}.

Table 3.1: Gain Ratios for Varying Input Sets.

	$S = \{PO, TPP, TPPc, EQ\}$	$S = \{DC, EC, PO, TPP, TPPc, EQ\}$
BndBnd	0.0	0.3359
IntBnd	1.0	0.6309
BndInt	1.0	0.5510
ExtBnd	1.0	0.5510
BndExt	1.0	0.5510

	$S = \{EC, PO, TPPc, NTPPc\}$
BndBnd	0.2619
IntBnd	0.2619
BndInt	0.2619
ExtBnd	1.0
BndExt	0.0

In the first two cases, we see that calculating the BndBnd intersection predicate gives us the least information. However in the first case, that calculation, in fact, would be wasted effort, as it would result in no new information. The third input set shows that not only would calculating BndExt be wasted effort, but ExtBnd

gains us the most information, as it has a significantly higher gain ratio than the other predicates. Based on the results for the first example, we would calculate any predicate except BndBnd; BndBnd would give us no new information, and in fact is a scenario where having a modification that allows us to bias the gain ratio for faster algorithms/implementations would be useful. In the second example, the IntBnd predicate would be selected for calculation. The ExtBnd predicate would be chosen for calculation in the third example. This sample input set also shows a scenario in which we have a definitively more useful predicate, but also a predicate that would be wasted effort (BndExt).

3.2. COMPLEXITY CONSIDERATIONS

The complexity of calculating any of the intersection predicates is dependent on many factors, not the least of which is the specific implementation. For example, our implementation of BndBnd uses trees of Axis Aligned Bounding Boxes (AABBs) to narrow down the number of triangular faces between the objects that could possibly intersect before resorting to computing triangle-triangle intersections [12]. Because of this, the worst case complexity for this predicate is $O(f_A * f_B)$, where f_A and f_B are the number of faces in the respective objects A and B. This only happens if the objects are situated such that all Axis Aligned Bounding Boxes in the objects overlap. The best case is if none of the Axis Aligned Bounding Boxes in the tree overlap, in which case we have a constant time comparison that is not dependent on the number of faces.

Hence, because the complexity of these calculations depends on the face counts of the objects and the relative orientation, the cost of calculating the next predicate to use is negligible compared to the cost of calculating the predicate.

4. EXPERIMENTAL TIMING AND RESULTS

4.1. EXPERIMENTAL SETUP

We timed the execution of our implementation of three predicates: BndBnd, BndInt, BndExt. It has been shown [5] that implementing these three predicates is sufficient to uniquely determine the RCC-8 relation of two objects because some predicates can be implemented with respect to others; IntBnd(A,B) == BndInt(B,A) and ExtBnd(A,B) == BndExt(B,A). The experiments were run 100 times on each of 44 model files that contained two 3D spheres in various configurations. Each sphere consisted of approximately 2000 faces; the execution time of the predicates depends mainly on the relative configuration of the spheres. File IO and internal representation generation were ignored across all executions of the predicates. Timing was performed on an AMD Bulldozer processor running at 3.1 GHz, with 12 GB of RAM available. Table 4.1 shows the results of the timings.

Table 4.1: Timing Statistics for Predicate Calculations.

	BndBnd	BndInt	BndExt
Average (sec)	0.0103	6.66	0.188
Min (sec)	0.00000431	0.000433	0.000459
Max (sec)	0.849	71.8	1.17
StdDev	0.2170442	17.846823	0.261670587

The complexity of the predicate chooser is dependent on the number of relations in the input set. For every execution of the predicate selector, a random subset of the RCC-8 relations with cardinality $1 < k \leq 8$ was chosen out of all possible subsets of RCC-8. The timings reported are the average of 100 executions of the predicate selector.

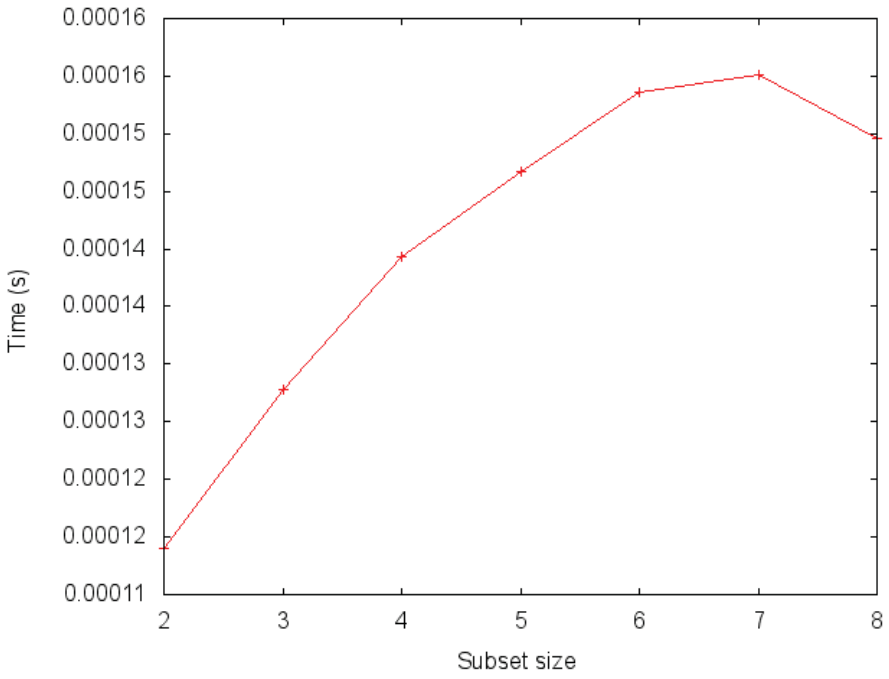


Figure 4.1: Predicate Selection Runtime vs. Input Subset Size.

4.2. RESULTS

Table 4.1 shows some statistical analysis of the runtimes of the three implemented intersection predicates. Figure 4.1 shows the runtime of the predicate selection algorithm for varying sized subsets of the RCC-8 relations. Selecting a predicate takes on average 1/100th of the time it takes to calculate the predicate itself. Using this approach avoids unnecessary calculations, and also allows us to bias calculations that are more efficient and precise on a given system. Currently, BndBnd is the most precise and efficient algorithm based on the data representation we use in our implementation. This ability to bias the selection allows us to avoid the most computationally expensive algorithms unless absolutely necessary.

5. ADDITIONAL OPTIMIZATIONS: MEMOIZATION

5.1. DEFINITION

Memoization is an optimization technique in which a function‘s result for a unique set of inputs is cached. When the function is called with the inputs again, the cached results are reused and the computation is avoided. This approach has two tradeoffs: memory usage and the overhead associated with checking the cache key. If checking a cache key is computationally negligible compared to the computation, the speedup is significant. Part of the overhead of checking the cache key is associated with the time it takes to access the cache: if the cache is too large to fit in memory, then cache hits become expensive. The tradeoffs between memory and speed must be weighed against each other. In our application, it was decided that the relatively small maximum number of possible cache keys (247) (namely, the number of possible subsets of RCC-8 relations) made the cost of additional memory negligible when compared to the theoretical magnitude of the speedup.

5.2. EXPERIMENTAL SETUP

The effect of memoization on the predicate selection algorithm was tested by executing the algorithm 10^n times, for $0 \leq n \leq 7$. Between each set of executions, the cache for the memoizer was cleared so that solutions for keys would have to be regenerated. For each input n, the experiment was run 100 times. The runtime reported for each input is an average over all runs for that parameter value.

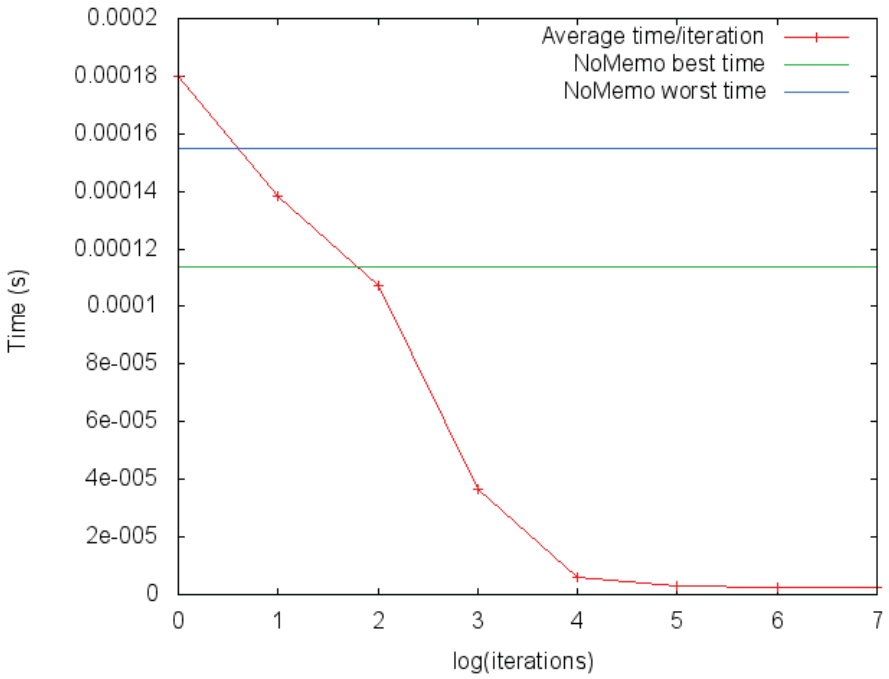


Figure 5.1: Memoized Predicate Selection Runtime per Number of Iterations.

5.3. RESULTS

Figure 5.1 shows that as the memoizer is allowed to run for more iterations, we reach the overhead of checking a cache key. Using memorization for a small number of iterations is more expensive then no memorization at all due to the overhead of checking the cache. After several iterations, the cost of checking the cache key is negligible compared to choosing the next predicate, resulting in more than a 100x speedup. It is worth noting that these timings were determined as if every subset of RCC-8 was possible. As a small number of these subsets can actually occur in the composition table, the chance of a cache miss will go to zero significantly faster, and as such the runtime will reach the time of checking the cache key in fewer iterations.

6. FUTURE WORK

Shortening the feedback loop is vital to efficient analysis of three dimensional data. Eliminating unnecessary calculations is just one step in achieving this goal. Future efforts will focus on further optimizing the calculations involved in the intersection predicates by investigating more efficient intersection algorithms and exploiting the resources available on modern computers, including general purpose computing on graphics cards and distributed computing.

7. CONCLUSIONS

In order for Qualitative Spatial Reasoning applications to be both feasible and useful, the information feedback loop between the computational engine and the user must be made as efficient as possible. Herein we presented a novel approach that combined the use of composition tables and decision trees to efficiently determine the spatial relation between two objects. Specifically, we showed how to leverage the benefits of calculating the intersection predicate with the highest informational gain. We also showed that the cost of caching the decision tree information greatly increases the efficiency of the predicate selection even at the cost of requiring additional memory. As the proliferation of 2D and 3D datasets continue, we hope that this work facilitates the implementation of other spatial reasoning systems in the future.

References

[1] Julia Albath, Jennifer L. Leopold, and Chaman L. Sabharwal. Visualization of Spatio-Temporal Reasoning Over 3D Images. In *Proceedings of the 2010 International Workshop on Visual Languages and Computing (in conjunction with the 16th International Conference on Distributed Multimedia Systems)*, DMS '10, pages 277–282, 2010.

[2] Julia Albath, Jennifer L. Leopold, Chaman L. Sabharwal, and Anne M. Maglia. RCC-3D: Qualitative Spatial Reasoning in 3D. In *Proceedings of the 23rd International Conference on Computer Applications in Industry and Engineering*, pages 74–79, November 2010.

[3] Eliseo Clementini, Jayant Sharma, and Max J. Egenhofer. Modelling topological spatial relations: Strategies for query processing. *Computers and Graphics*, 18(6):815 – 822, 1994.

[4] Anthony G. Cohn, Brandom Bennett, John Gooday, and Micholas Mark Gotts. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *Geoinformatica*, 1(3):275–316, October 1997.

[5] Nathan Eloe, Jennifer L. Leopold, Chaman L. Sabharwal, and Zhaozheng Yin. Efficient Computation of Object Boundary Intersection and Error Tolerance in VRCC-3D+. In *Distributed Multimedia Systems '12*, 2012.

[6] A. Karlsson. GIS and Spatial Extensions with MySQL. *MySQL Developer Zone, http://dev. mysql. com/tech-resources/articles/4.1/giswith-mysql. html*, 2007.

[7] A. Murta. A General Polygon Clipping Library. *Advanced Interfaces Group, Department of Computer Science, University of Manchester. On-line resource. URL: http://www. cs.-man. ac. uk/aig/staff/alan/software/gpc. html*, 2000.

[8] Jihong Ouyang, Qian Fu, and Dayou Liu. A Model for Representing Topological Relations Between Simple Concave Regions. In *Proceedings of the 7th international conference on Computational Science, Part I: ICCS 2007*, ICCS '07, pages 160–167, Berlin, Heidelberg, 2007. Springer-Verlag.

[9] J.R. Quinlan. Induction of Decision Trees. *Machine learning*, 1(1):81–106, 1986.

[10] J.R. Quinlan. *C4. 5: Programs for Machine Learning*, volume 1. Morgan kaufmann, 1993.

[11] David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions and Connection. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Proceedings of the 3rd Conference on Principles of Knowledge Representation and Reasoning*, KR '92, pages 165–176. Morgan Kaufmann, San Mateo, California, 1992.

[12] C. Sabharwal. Survey of Implementations of Cross Intersection Between Triangular Surfaces. MDC Report Q0909 (Now Boeing at St. Louis, MO-USA), 1987.

[13] Chaman Sabharwal and Jennifer L. Leopold. Reducing 9-Intersection to 4-Intersection for Identifying Relations in Region Connection Calculus. In *The 24th International Conference on Computer Applications in Industry and Engineering*, pages 118–123. CAINE, November 2011.

[14] Chaman Sabharwal and Jennifer L. Leopold. Smooth Transition Neighborhood Graphs for 3D Spatial Relations. In *2013 IEEE Symposium Series on Computational Intelligence*, 2013.

[15] Chaman L. Sabharwal, Jennifer L. Leopold, and Nathan Eloe. A More Expressive 3D Region Connection Calculus. In *Proceedings of the 2011 International Workshop on Visual Languages and Computing (in conjunction with the 17th International Conference an Distributed Multimedia Sysems (DMS 11))*, pages 307–311, August 2011.

[16] Sandro Santilli, Mark Leslie, Chris Hodgson, Paul Ramsey, Jeff Lounsbury, and Dave Blasby. PostGIS Manual. *Refractions Research Inc*, 2005.

[17] M. Stocker and E. Sirin. PelletSpatial: A Hybrid Region Connection Calculus RCC-8 and RDF/OWL Reasoning and Query Engine. *Proceedings of Web Ontology Language (OWL): Experiences and Directions 2009 (OWLED 2009)*, 2009.

[18] D.W.S. Wong and J. Lee. *Statistical Analysis of Geographic Information With ArcView GIS and ArcGIS*, volume 1. Wiley, 2005.

III. Spatial Temporal Reasoning Using Image Processing, Physics, and Qualitative Spatial Reasoning

Nathan Eloe *Jennifer L. Leopold* *Chaman L. Sabharwal*

nwe5g8@mst.edu *leopoldj@mst.edu* *chaman@mst.edu*

ABSTRACT

Qualitative spatial reasoning (QSR) is a powerful tool in automated computer reasoning, a necessary step forward in fields like computer vision and media analysis. Stereographical multimedia has rapidly become a prevalent part of technological culture, and the amount of these kinds of data that exists is staggering. Humans interpret depth information using prior knowledge that a computer lacks. This prior knowledge stems from remembered observance of the basic laws of physics. While the computer lacks the intuitive understanding of these principal physical properties, it is capable of determining more precise information through faster calculation. Herein the authors explore the information that can be gained from an amalgamation of QSR methods and physics, and present some preliminary results from an implementation based on this powerful combination.

1. INTRODUCTION

Human perception of three dimensions is a complex field. A person determining the shape of an object must rely heavily on visual cues in the form of lighting, shape, and depth information [18]. Where this information is insufficient to unambiguously identify an object, the observer must make a judgment that may not be consistent with that of all other observers [18]. Research has shown that an individual's desires can influence the way things are perceived ("Wishful Seeing") [7].

This raises some important questions in the field of computer vision, including:

- How can the computer deal with ambiguous visual information?
- With no base of previous experience, what information should the computer use to reduce ambiguity?
- How can the information gain of the system be maximized while the computational cost is minimized? In other words, what calculations should be done to obtain the most information with the least work?

With stereoscopic media becoming ubiquitous in the form of 3D movies and consumer electronics (e.g. televisions and portable gaming devices), there is a growing, urgent need for computational analysis of these data. Such technology could have impact in physical security (e.g. analysis of images from multiple sources such as security cameras), robotic vision, and defense (e.g. identification of potential dangers and suspicious behavior from stereoscopic information).

Image processing techniques can provide insight into a system recorded stereoscopically, but only about what can be seen by the cameras. Humans use experience and prior knowledge to make assumptions about parts of the scene that are hidden from view. One example of this is a speaker behind a podium; an audience member

would know that a human standing behind a podium most likely has legs, and that the podium does not continue back into infinity because the objects in the scene (human, podium) are known quantities that have been encountered before. The computer does not have this same experience; image processing techniques alone would only allow the computer's camera to know about what is directly visible.

In this paper, the authors explore the use of Qualitative Spatial Reasoning (QSR) methods and basic physical properties in addition to visual information from the scene to reduce the amount of incomplete visual information. Spatial information gained from QSR and physics is retroactively applied to the scene to further reduce ambiguity; present knowledge about a system is used to revise past assumptions, which improves the precision of current and future data. In section 2, an introduction to image processing, QSR, and physical properties is presented. Section 3 describes the constrained experimental system. Section 4 contains initial experimental results, while Sections 5 and 6 describe the evolution of the system and the results the system modifications. Conclusions and future work are discussed in Section 7.

2. BACKGROUND AND RELATED WORK

2.1. IMAGE PROCESSING AND DISPARITY

Image processing is an important field in computer and robotic vision. A significant amount of research in this area has been devoted to finding computationally efficient algorithms; images are inherently two dimensional, which implies that most naive algorithms are at best $O(m \times n)$ in their computational complexity for an $m \times n$ image. The persistence of high resolution images (full high definition already being common and 4k resolution beginning to emerge) means that these algorithms will be computationally expensive. Many image formats are 4-channel (giving an

$m \times n \times 4$ data structure size to hold RGBA or HSVA (Hue Saturation Value Alpha) information, two popular information formats), which only serves to incresae the amount of computation needed on a single image.

Disparity [3, 9, 6] and the parallax effect are two concepts exploited in image processing to mimic human perception of depth; objects closer to the observer appear larger than more distant objects. Thus, by determining the parallax between occurrences of an object in each of a pair of stereo images, the relative distance from the cameras to the object can be determined. Disparity also has been used to estimate the motion of objects [6]. It is an invaluable tool in determining spatial information from multiple observations of the same scene.

2.2. HUMAN PERCEPTION IN THREE DIMENSIONS

Human 3D perception is fascinating: by all reckoning, such a feat should be mathematically impossible with the abstract data the brain receives from the eyes [18]. Regardless, humans are capable of making relatively consistent judgments about shapes and motion in three dimensions using only data from two "cameras" (the eyes) and a base of experience. Learned behavior such as object permanence [2] show that prior knowledge is required to make judgments about three dimensional space. Mimicking human perception with a computer is an important facet of computer and robotic vision.

2.3. QUALITATIVE SPATIAL REASONING (QSR)

Qualitative Spatial Reasoning (QSR) has varying applications in Geographic Information Systems (GIS), visual programming language semantics, and digital image analysis [11, 5, 10, 14]. Systems for spatial reasoning over a set of objects have

evolved in both expressive power and complexity. The design of each system focuses on certain criteria, including efficiency of computation, ease of human comprehension, and expressive power.

The spatial reasoning system chosen for this investigation is VRCC-3D+ [15], an expansion and implementation of the RCC-3D [1] system designed by Albath et al. As opposed to other RCC systems (many of which have no implementation), the relations in VRCC-3D+ express both connectivity (in 3D) and obscuration. Obscuration will change from viewpoint to viewpoint, but connectivity is a global property that can be used to discern new information at every perspective in the system.

For this work, the authors focus on the obscuration element of the VRCC-3D+ relation. The connectivity portion of the relation will become important as the system is expanded to handle an arbitrary number of cameras and vantage points. VRCC-3D+ identifies four basic kinds of obscuration: no obscuration ($nObs$), partial obscuration ($pObs$), complete obscuration ($cObs$), and equal obscuration ($eObs$). The system further breaks each base obscuration into three different classes: regular obscuration (object A obscures object B), converse obscuration (object A is obscured by object B), and equal/mutual obscuration (object A and object B obscure each other). At this point in the investigation, this further classification is unimportant; it only matters if obscuration is present between two objects, not which object is being obscured.

2.4. INERTIA AND CONSERVATION OF MASS AND ENERGY

There are a multitude of physical properties that can be used to discern information about spatial relationships. Every property used to derive spatial information introduces a new computational cost and has an upper limit to the amount of information it can deduce. The ideal property would be one that would give insight into the system without requiring any new calculation. When this is impossible, the goal

should be to maximize the ratio of information gain to computational cost. One of the goals of this research is to discover a combination of physical properties that maximizes this ratio.

As a starting point, two physical properties will be examined: inertia and conservation of mass and energy. Inertia is best described by its colloquial definition: an object at rest tends to stay at rest, and an object in motion tends to stay in motion. More formally, inertia is the resistance a physical object has against a change in its state of motion or rest. This can provide useful insight into the physical relationship of two objects. Given two objects, if one passes behind another, it can be used to determine whether or not the objects collided at any point. In terms of spatial connectivity, this collision will correspond to a change from a disconnected (DC) state to an externally connected (EC) state. This in turn gives useful information because it defines a known point on the (possibly hidden) boundary of one or both objects.

Conservation of mass and energy will also be used in conjunction with inertia to gain additional information. If an object becomes obscured by another object, its trajectory can be estimated. If the actual trajectory is different than the calculated trajectory, then something must have changed the state of motion or rest of one or both objects. Using the difference in expected and actual position at a given time to revise earlier calculations results in a corrected physical model that yields additional information about the entire system.

2.5. CURRENT WORK IN QUALITATIVE SPATIAL AND TEMPORAL REASONING

Qualitative spatial and temporal reasoning has been an active field in recent years. Takahashi [17] explored using a new expansion to RCC-8 in which he uses two specific vantage points such that the lines of sight are perpendicular. Connectivity and obscuration were determined from each location to give a more precise determination

about the objects in the scene. Takahashi's work differs from this work in that he uses a front and bird's eye ("side" and "upper") view to obtain information. In contrast, this work focuses on emulating human sight using stereo images, which will be expanded to include information from additional visual sources.

Renz [12] proposed efficient algorithms for determining tractable subsets of RCC-8 and the Interval Algebra by phrasing the problem as a consistency satisfaction problem CSPSAT(S) and refining the relation set when necessary. Directly determining the relations between objects in space and time is not a direct consequence of these tractable subsets, but any reduction in the size of the subset of possible relations can substantially increase the efficiency of determining actual relations between objects [8]. These tractable subsets can be used to aid in the disambiguation of information from multiple sources and will be exploited in this research.

Renz and Ligozat [13] performed a theoretical analysis of spatial temporal reasoning systems and showed that if a system exists such that weak composition does not result in actual composition, path consistency no longer applies. In these cases, algebraic closures of the system must be used to determine composition. They examine the effects of weak composition on spatial temporal reasoning systems and provide a methodology to analyze spatial and temporal calculi. While purely theoretical, this work benefits qualitative spatial and temporal reasoning. Path consistency and composition are two important attributes of a QSR system that have been exploited to aid in automated reasoning; analysis of this work to show these facets of spatio-temporal reasoning are not violated will be important to the continued usefulness of the system.

Ye and Hua [19] explored using depth cameras to determine three dimensional spatial relations. They did not apply their work to a series of images over time, and use specialized depth finding cameras to determine depth (the Xbox Kinect). As the research presented in this paper is expanded to include additional visual sources, Ye and Hua's work may be investigated as another kind of information source.

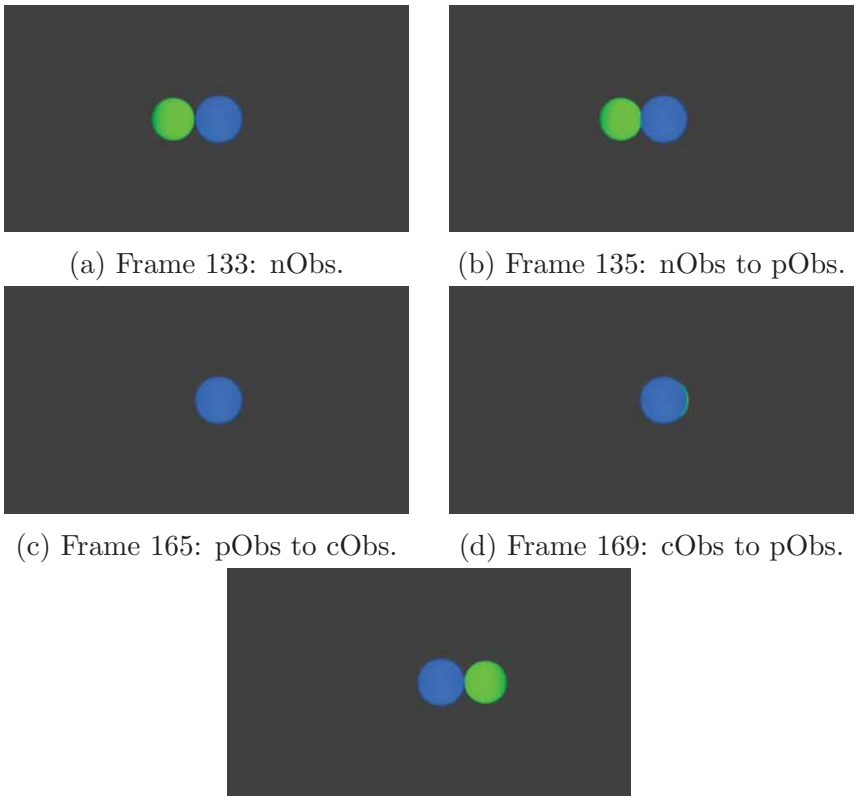
In 2007, Santos [16] investigated a framework in which the depth and motion of an object may be used in reasoning while accounting for the observer's viewpoint. He presented a formal logic based approach to reasoning about depth and motion that he used in a robotic vision application called the Depth Profile Calculus (DPCC). DPCC uses depth maps obtained through disparity calculations to determine information about three dimensional space, but ignores many other visual cues available (such as color, lighting, and other physical properties). In this work, the authors use similar methods, but incorporate additional information to get a more accurate view of the world.

3. COMPUTATIONAL SPATIAL AND TEMPORAL REASONING

As an initial exploration, the authors constrained the system of interest as follows:

- The system is modeled as a single rolling green sphere that passes behind a stationary blue sphere but does not collide (Figure 3.1).
- The system is simulated using Blender 2.64 [4] with two separate camera positions to guarantee that frames from the cameras would be showing different perspectives of the same point in time.
- The cameras were aligned such that the direction of views were parallel and the top row of the left camera's image corresponded to the top row of the right camera's image. This differs from human vision slightly, as the computer does not need to "focus" on an object by pointing both cameras at it; its visual information is more complete than a human's over the entire image.
- The floor of the system was transparent.

- The moving sphere's trajectory was perpendicular to the view direction of the cameras.



(a) Frame 133: nObs.

(b) Frame 135: nObs to pObs.

(c) Frame 165: pObs to cObs.

(d) Frame 169: cObs to pObs.

(e) Frame 196: pObs to nObs.

Figure 3.1: Images From Analyzed Video: as Seen From the Left Camera. The green sphere is further from the cameras than the blue sphere, and as such appears smaller.

The following constraints were placed on the system to allow simplifications that are considered to be unimportant in the context of this work:

- Masking the image using HSV (Hue Saturation Value) values was used for image segmentation into objects.
- Disparity was calculated for each object by finding the center of the matching bounding box of objects and determining the difference in the x direction.

Analysis of stereoscopic videos is a three step process: *frame analysis*. *obscuration analysis*, and *object analysis*.

3.1. FRAME ANALYSIS

In the context of this work, a *frame pair* is a pair of stereo images from a left and right oriented camera that portray the same moment in time from different perspectives. For every frame pair in the videos the following actions are taken:

- The images are converted from the default representation to HSV.
- Range filtering is used to determine the locations of both objects in the images.
- The disparity and bounding rectangle are calculated for each object.
- The bounding rectangle and disparity for each object are stored, along with the frame number.

3.2. OBSCURATION ANALYSIS

For this paper, obscuration and object analysis occur with respect to the left camera. The results could be refined by using information from both cameras.

The following pseudocode is used to determine the obscuration from the left camera at every step. The list of bounding rectangles and disparities from the frame analysis is stored in steps. The green sphere corresponds to object A in the pseudocode, the blue sphere is object B, and bbox refers to an object's bounding box.

```
obss = [] #the list of obscurations
for s in steps:
  if object A has a bbox in s:
    xa = A's bbox x location in s
```

```
        wa = A's bbox width in s
        if object B has a bbox:
          xb = B's bbox location in s
          wb = B's bbox width in s
          if the bboxes overlap:
            lastO = 'pObs'
          else:
            lastO = 'nObs'

        else:
          lastO = 'cObs'
      else:
        lastO = 'cObs'
      obss.append(lastO)
```

The eObs obscuration type is combined with cObs; not enough information exists in this experiment to distinguish between cObs and eObs.

Note that there is no distinction as to which object obscures the other, just that some obscuration occurs. It was visually verified that this code correctly identified changes in obscuration with respect to the left camera's video feed. Figure 3.1 shows the frames identified as changes in obscuration occurred from the left camera's perspective.

3.3. OBJECT ANALYSIS

In the object analysis step, the positions and depths of each object are determined. Position is determined using the right most edge of the bounding box. If no obscuration is detected from either perspective, the depth and position of the object

are directly recorded. Otherwise a polynomial is fit to the previous values recorded and used to estimate the current location. Every direction of movement (x, y, z) is handled independently. Due to the simplicity of the nature of this system, a linear fit was used; as the system is generalized, the order of the polynomial can be increased to handle differing kinds of acceleration and forces.

4. EXPERIMENTAL RESULTS

Figure 4.1 shows the positions of objects from a bird's eye view of the system. Every marker on the graph shows an observed or estimated object location of a particular object. This information can be remarkably helpful in learning about the structure of the system. For example, it may be possible to determine from using only the stereo images that the blue sphere does not extend into infinity due to the perspective nature of the projections. However, depending on the intrinsic properties of the camera, there could be a large area in space that may or may not contain the blue sphere. Using the information gained from projecting the path of the green sphere behind the blue sphere, it can be concluded that the green sphere did not collide with the blue sphere, so an upper bound is placed on how far back the blue object can extend.

This figure illustrates that this line of inquiry shows promise: a relatively accurate extrapolation of the green ball's location is feasible with a relatively small number of data points. This estimation could be improved further by including the observed position of the green ball in later frames, then using that information to retroactively correct the estimations of the location of the ball. This will allow information inferred from that estimation to be refined even further.

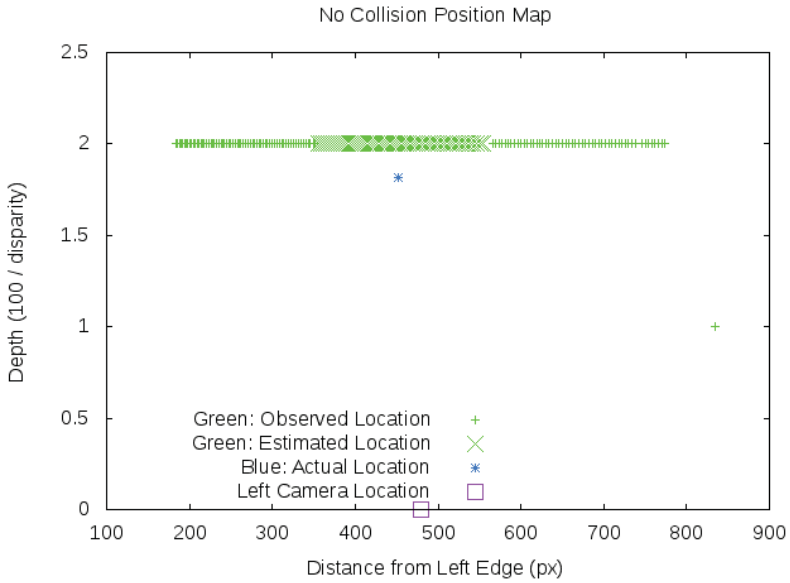


Figure 4.1: Observed and Extrapolated Positions. Each data point is a frame. Motion is from left to right.

5. THE MOVE TO DETECTORS AND COLLISION DETECTION

The implementation described has two severe limitations: a significant amount of preprocessing is required (in the form of frame and obscuration analysis) and it is resistant to expansion and refinement. As such, the architecture was redesigned to use *detectors*.

5.1. DETECTORS: THE BASIC BUILDING BLOCK

In the scope of this work, a detector is defined as a function or functor that takes as parameters the object image masks from each camera and a history of locations for each object. The action the detector takes depends on the purpose of the

detector. A motion detector might update the location history of each object. An obscuration detector could update an internal state reflecting the current obscuration present between two objects. Analysis of videos using this detector architecture addresses the major downfalls of the initial implementation. Analyzing stereo videos now follows these steps:

```
initialize detectors (motion, obscuration, ...)
initialize object location history
while stopping criteria not met:
  generate object masks
  for each detector:
    apply detector and handle emitted signals, if necessary
```

By making the detectors self-contained and independent, this system is easy to modify and maintain.

5.2. COLLISION DETECTION

As a proof of concept of this architecture, a simple collision detector was implemented. This detector works on the principle of inertia; an object in motion tends to stay in motion, while an object at rest tends to stay at rest. At a given time step t, if an object's position at times $t-1$ and $t-2$ were identical, but the position at time t was different than at $t-1$, a collision was reported to the user. For now, only motion in the x dimension determines collision, although this will be expanded to other dimensions in the future.

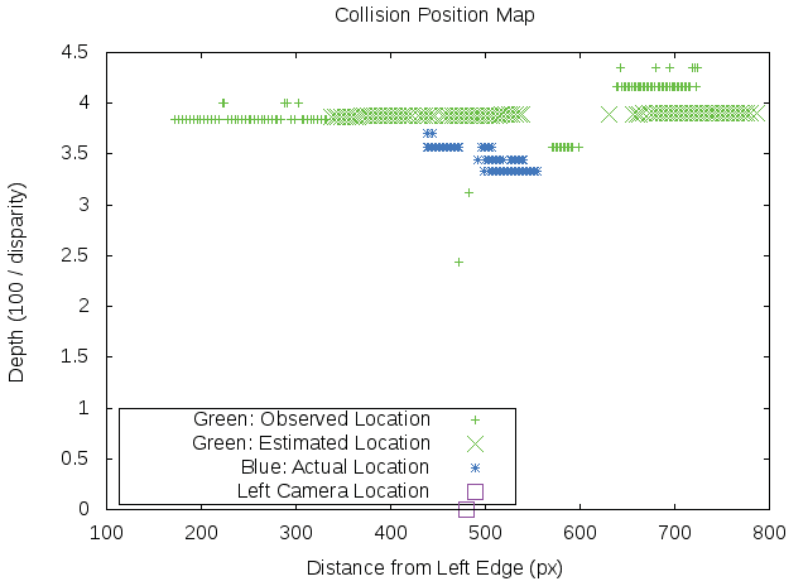


Figure 6.1: Observed and Extrapolated Positions. Each data point is a frame. Motion is from left to right.

6. RESULTS WITH COLLISION AND DETECTORS

Figure 6.1 shows a position map for a video where collision occurs between the green and blue balls, causing the blue ball to roll to the right side of the scene. This position map is not as clear as Figure 4.1, but the anomalies can be explained. The estimated positions for the green ball extend far past where the calculated points begin again. Physically this makes sense; when the green ball collides with the blue ball, some kinetic energy is transferred to the blue ball, causing the velocity of the green ball to decrease. Because the estimations use the velocity as calculated before collision, the final estimated position of the green ball is significantly different than observed. This is a phenomenon that will facilitate a retroactive learning detector; information like this can teach a great deal about the shapes of the objects in the system.

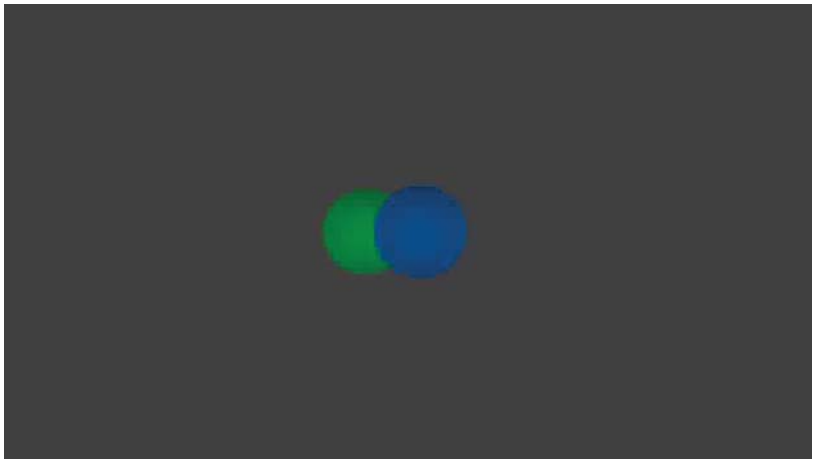


Figure 6.2: Frame at Which the Detector Noted Collision.

The depth appears in striations because the disparity is an integer number of pixels, and the calculated depth will be discrete values, not a continuous variable. It is for this reason the depth was not initially used to determine collisions. Also, as the depth of an object changes, the observed x coordinates do not behave as expected. This is due to the number of degrees of freedom in the outline of an object. As objects travel across the scene, a single point of reference is needed to determine its location. The center of mass of the object is a possibility, but as the objects obscure each other, the perceived center of mass changes. Using the leading or trailing edge of the bounding box around the object works until the object is obscured as well. Also, as objects get closer to the camera, the object will appear bigger, changing both the center of mass and the leading and trailing edges of the bounding box. This poses an interesting open research question: can there be a reliable point of reference in an object that can be used to describe its position in 3-space?

The collision detector flagged a collision between the objects at the frame shown in Figure 6.2. This collision was visually verified.

7. CONCLUSIONS AND FUTURE WORK

Using physical properties in conjunction with QSR and image processing methods is a promising direction in the field of computational vision and spatio–temporal reasoning. This could have applications in physical security (automated CCTV analysis), media analysis, and many other multimedia fields.

In this paper, the authors have initiated an exploration into using these three areas to accomplish automated spatio–temporal reasoning. The results of this initial research are encouraging. This work will be continued to allow analysis of systems with fewer constraints, consider additional physical properties, and eventually be applied to video feed of live events from cameras, not just rendered static physical simulations. Different combinations of physical properties and image processing techniques also will be investigated to find a high information gain to computational cost ratio.

References

[1] Julia Albath, Jennifer L. Leopold, Chaman L. Sabharwal, and Anne M. Maglia. RCC-3D: Qualitative Spatial Reasoning in 3D. In *Proceedings of the 23rd International Conference on Computer Applications in Industry and Engineering*, pages 74–79, November 2010.

[2] Renee Baillargeon. Representing the Existence and the Location of Hidden Objects: Object Permanence in 6- and 8-month-old Infants. *Cognition*, 23(1):21–41, 1986.

[3] Stephen T. Barnard. Disparity Analysis of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):333–340, July 1980.

[4] Blender Foundation. blender.org - Home. `http://blender.org`, April 2013.

[5] Anthony G. Cohn, Brandom Bennett, John Gooday, and Micholas Mark Gotts. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *Geoinformatica*, 1(3):275–316, October 1997.

[6] D. Demirdjian and T. Darrell. Motion Estimation from Disaprity Images. In *Proceedings of the Eighth IEEE International Conference on Computer Vision ICCV*, volume 1, pages 213–218, 2001.

[7] David Dunning and Emily Balcetis. Wishful Seeing: How Preferences Shape Visual Perception. *Current Directions in Psychological Science*, 22(1):33–37, February 2013.

[8] Nathan Eloe, Jennifer L. Leopold, Chaman L. Sabharwal, and Douglas McGeehan. Efficient Determination of Spatial Relations Using Composition Tables and Decision Trees. In *Proceedings of the IEEE Symposium on Computational Intelligence for Multimedia, Signal, and Vision Processing*, CIMSIVP '13, pages 1–7, 2013.

[9] Karsten Mühlmann, Dennis Maier, Jürgen Hesser, and Reinhard Männer. Calculating Dense Disparity Maps from Color Stereo Images, an Efficient Implementation. *International Journal of Computer Vision*, 47(1–3):79–88, April 2002.

[10] David A. Randell and Mark Witkowski. Using Occlusion Calculi to Interpret Digital Images. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *Proceedings of the 17th European Conference on Artificial Intelligence*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 432–436. IOS Press, August 2006.

[11] Jochen Renz. *Qualitative Spatial Reasoning with Topological Information*. Springer-Verlag, 2002.

[12] Jochen Renz. Qualitative Spatial and Temporal Reasoning: Efficient Algorithms for Everyone. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 526–531, January 2007.

[13] Jochen Renz and Gérard Ligozat. Weak Composition for Qualitative Spatial and Temporal Reasoning. In Peter Beek, editor, *Principles and Practice of Constraint Programming – CP 2005*, volume 3709 of *Lecture Notes in Computer Science*, pages 534–548. 2005.

[14] Pascal Rost, Lothar Hotz, and Stephanie von Riegen. Supporting Mobile Robot's Tasks through Qualitative Spatial Reasoning. In *Procceedings of the Ninth International Conference on Informatics in Control, Automation and Robotics (ICINCO 2012)*, volume 2, pages 394–399, 2012.

[15] Chaman L. Sabharwal, Jennifer L. Leopold, and Nathan Eloe. A More Expressive 3D Region Connection Calculus. In *Proceedings of the 2011 International Workshop on Visual Languages and Computing (in conjunction with the 17th International Conference an Distributed Multimedia Sysems (DMS 11))*, pages 307–311, August 2011.

[16] Paulo E. Santos. Reasoning about Depth and Motion from an Observer's Viewpoint. *Spatial Cognition and Computation: An Interdisciplinary Journal*, 7(2):133–178, December 2007.

[17] Kazuko Takahashi. Reasoning about Relative Relationships in 3D Space for Object Extracted from Dynamic Image Data. In *Proceedings of the 2012 ECAI Workshop on Spatio–Temporal Dynamics*, pages 45–51, August 2012.

[18] James T. Todd. The Visual Perception of 3D Shape. *TRENDS in Cognitive Sciences*, 8(3):115–121, March 2004.

[19] Jun Ye and Kien A. Hua. Exploiting Depth Camera for 3D Spatial Relationship Interpretation. In *Proceedings of the Multimedia Systems Conference*, February 2013.

IV. Dual Graph Partitioning For Bottom-Up BVH Construction

Nathan W. Eloe *Joseph A. Steurer* *Jennifer L. Leopold*

nwe5g8@mst.edu *asjxc9@mst.edu* *leopoldj@mst.edu*

Chaman L. Sabharwal

chaman@mst.edu

ABSTRACT

Bounding Volume Hierarchies (BVHs) are essential tools in performing collision detection on three-dimensional information. They reduce the number of expensive calculations required to determine whether or not two geometrical entities collide by using inexpensive calculations to rule out parts of the objects that could not possibly intersect. Quickly producing a high quality BVH is an important aspect of three-dimensional multimedia analysis. As such a powerful optimization, efficient and high quality BVHs are still an active area of research. Herein, the authors present a novel BVH representation that reduces the redundancy in the tree structure by allowing a node to contain an arbitrary number of children, as well as compressing non-unique nodes and combining their children. A new partitioning scheme using a graphical representation of the object is also presented to improve the quality of the generated BVH.

Keywords Bounding Volume Hierarchies, Dual Graph, Axis Aligned Bounding Box, Qualitative Spatial Reasoning

1. INTRODUCTION

The current trend toward big (multimedia) data analysis necessitates the ability to quickly process and analyze the vast amount of three dimensional information that is being generated. Collision detection between rendered static objects is a computationally complex process; even performing a ray-casting collision check can be a time-consuming process if steps are not taken to optimize the number of calculations. A commonly used mechanism to significantly optimize the computation time is a Bounding Volume Hierarchy (BVH) [14]. A BVH subdivides a portion of space into smaller volumes containing objects of interest. Each sub-volume is then adaptively divided until some atomic level is reached. The BVH is used to efficiently pare out parts of a volume space that could not possibly contribute to the intersection query, resulting in the removal of a large number of potentially expensive calculations.

A BVH is commonly created in one of three different ways: Top-Down, Bottom-Up, or Iterative Insertion. Each of these generation mechanisms has strengths and weaknesses relating to the creation time and the quality of the resulting BVH [10]. One BVH is considered to be of higher quality than another BVH if collision queries can be performed faster. As such, a higher quality BVH tends to minimize the total volume contained in the bounding volumes and be as compact as possible. This is roughly analogous to the *Surface Area Heuristic* (SAH), a mechanism used to determine the expected cost to perform a ray trace [11], though recently additional quality metrics on Bounding Volume Heuristics have been suggested [2]. Because collision detection is frequently used in determining the spatial relation between objects in

qualitative spatial reasoning systems such as VRCC-3D+ [13], the implementation of a BVH subsequently can play a pivotal role in a spatial reasoner's information feedback loop and the user's overall experience with an application.

Herein the authors investigate the use of a dual graph representation of a three-dimensional object with triangulated surface boundary to improve on the quality of a BVH generated by using a bottom-up approach. This approach is not designed to replace existing BVH creation algorithms; it should be considered as an enhancement that can be integrated into already existing algorithms that use other optimizations. An implementation of a BVH using Axis Aligned Bounding Boxes (AABB) as the bounding volume and a representation that removes internal redundancy is presented here for use in benchmarking the dual graph partitioning scheme.

The remainder of this paper is organized as follows. Section 2 reviews Bounding Volume Hierarchies and Dual Graphs. Section 3 introduces the internal structure of the AABB Tree used in the implementation of VRCC-3D+ that reduces redundancy and number of nodes. The Dual Graph partitioning scheme is presented in Section 4. Section 5 outlines the experimental setup; results from these experiments are presented in Section 6. Conclusions drawn from the results are presented in Section 7. Section 8 outlines the future work that will be undertaken.

2. BACKGROUND

2.1. BOUNDING VOLUME HIERARCHIES

BVHs are an efficient method for quickly handling ray intersection and collision detection in a three dimensional environment. BVHs can reduce the computation time of ray/collision detection logarithmically, as a child node in the hierarchy doesn't need to be calculated if a parent node is not in the intersection.

There are three kinds of commonly employed methods: Top-Down, Bottom-Up, and Iterative Insertion. Top-down is the most common approach as it is faster than naïve collision detection, easy to implement, and efficient to create [15, 10]. If performed correctly, a bottom-up tree generation is likely to produce a higher quality tree than a top-down approach [5]. Insertion methods are commonly used when not all objects in a scene are visible, as they allow objects to be added dynamically [6]. These approaches are based on the assumption that geometric primitives are predetermined. Trees are sometimes dynamically created based on the complexity and intersection outcome of the objects. In the surface-surface intersection, none of these methods is sufficient [7].

The Top-Down method works by wrapping a scene or an object in a bounding volume, typically the root of a tree. This bounding volume is subdivided into smaller volumes, with these volumes becoming the children of the root of the tree. This process is recursively performed until some stopping criterion is met. Common stopping criteria are minimum bounding volume, minimum number of primitives contained in a tree node, or maximum depth of the tree [9].

The Bottom-Up creation algorithm begins with the primitives and groups them, creating a bounding volume around each grouping. This process is repeated, treating each bounding volume as a new primitive. This process minimizes the unnecessary space contained within the internal nodes, and frequently produces a higher quality tree than a top-down method provided all the primitives are predetermined [16].

The third method, Iterative Insertion, begins with an empty tree, and inserts objects into the tree as they become visible. This allows for a dynamic scene to be processed using a BVH [9].

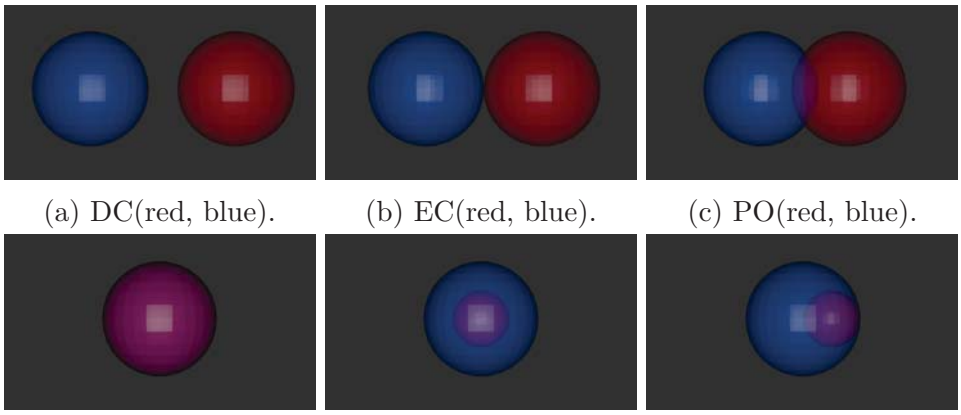
Herein the authors focus on a modification of the Bottom-Up approach. By using a dual graph (see Section 2.2) to represent a three-dimensional object, primitives can be grouped such that they are spatially close to each other, resulting in a more optimal tree than a naïve grouping. A novel representation of a BVH that reduces the internal size of the tree by removing redundancy is used as a benchmark; a more thorough examination of the implementation is presented in Section 3.

2.2. DUAL GRAPHS

In graph theory, the dual graph of graph $G = (V, E)$, is denoted as G^*. In $G^* = (V^*, E^*)$, a vertex $v^* \in V^*$ represents a face in G. An edge $e^* \in E^*$ exists between any two vertices v_1^* and v_2^* if the faces they represent in G share at least one edge [17].

A common representation of three-dimensional objects in CAD/CAM software is the ANSI B-rep model utilizing the triangulation of the boundary surface. The object boundary is specified as a mesh of vertices in space, with edges connecting those points, and triangular faces enclosed by the edges. This is directly analogous to a graphical representation of the object. By considering every vertex as a node in the graph, and edges between vertices as edges in the graph, the mesh representation of the object boundary becomes a graph. The triangles that comprise the approximation of the surface are called the faces of the object.

A dual graph directly follows; by allowing each triangular face in the object to be represented as a vertex in G^*, edges are defined in G^* as any two faces that share exactly two vertices $v_i, v_j \in V$ of $G(V, E)$. The dual graph allows visualization of how faces are arranged, and insights into the structure of the object can be gleaned from a simple breadth-first or depth-first search of the graph. In using the geometric Dual Graph to partition the faces of an object, the breadth-first search is used to form a spanning tree of half of the faces so as to keep the groupings as compact as possible.



(a) DC(red, blue). (b) EC(red, blue). (c) PO(red, blue).

(d) EQ(red, blue). (e) NTPP(red, blue) = NTPPc(blue, red). (f) TPP(red, blue) = TPPc(blue, red).

Figure 3.1: The VRCC-3D+ Connectivity Relations. The centers of all spheres lie in the same YZ plane; only the X coordinate changes.

3. BVH IN VRCC-3D+

VRCC-3D+ [3, 13, 12] is the implementation of a region connection calculus that qualitatively determines the spatial relations between three dimensional objects, both in terms of connectivity and obscuration. The VRCC-3D+ connectivity relations are calculated in three dimensions, and include: disconnected (DC), externally connected (EC), partial overlap (PO), equality (EQ), tangential proper part (TPP), non-tangential proper part (NTPP), tangential proper part converse (TPPc), and non-tangential proper part converse (NTPPc); see Figure 3.1. A composite VRCC-3D+ relation specifies both a connectivity relation and an obscuration relation. Obscuration is considered from a 2D projection and a depth relation. There are fifteen different obscuration relations defined in VRCC-3D+; each is a refinement of basic concepts: no obscuration, partial obscuration, and complete obscuration. For a more complete discussion of VRCC-3D+, see [3, 13, 12].

The implementation of VRCC-3D+ utilizes a large number of collision detection determinations to calculate the connectivity relationships between objects, and ray-casting techniques to determine how objects obscure one another from a given vantage point. If the size or location of an object in a scene changes, the object's spatial relationships with other objects in the scene may change. In turn, each recalculation of a VRCC-3D+ relation requires creation and/or queries of BVHs.

The VRCC-3D+ BVH implementation uses Axis Aligned Bounding Boxes (AABBs) as the bounding volume. The creation of an AABB is efficient, requiring only the minimum and maximum coordinate in each dimension. This operation is linear in the number of points that are being bounded if there is no known ordering of the points. Intersection tests involving AABBs are simple in their logic and execution for several types of primitives, including line segments, rays, and other AABBs. For these reasons, libraries such as CGAL [1] have AABB trees implemented as part of the standard set of tools used in computational geometry.

As with any BVH, the AABB has some drawbacks. There are many cases where the AABB might not be an optimal (tight fitting) bounding volume. Other bounding volumes, like Oriented Bounding Boxes [4] (OBBs) may give tighter bounds. However, their creation is more complex. Other bounding volumes, such as Bounding Spheres are easier to test for intersection with. Bounding spheres are easy to create, but the creation process is an approximation; it is unlikely that a Bounding Sphere will be tightly fitting around the points it encloses.

For the initial implementation, it was decided that the AABB was the best way to proceed. The simple creation of the bounding volume and the ease of representing the intersection calculations allowed for fast implementation and testing. The BVH in VRCC-3D+ is designed such that a different bounding volume can be used without

significant effort. Also, because the boxes are aligned along the global axes, multiple groupings of points are more likely to have identical bounding boxes. This property is exploited in the representation introduced below.

The BVH used in VRCC-3D+ is in the worst case a binary tree. Algorithm 1 shows the tree creation pseudocode; the nodes are partitioned into two sets, and then trees are created from the partitions. When implemented as a tree, the result is a binary tree. However, in the initial implementation, several redundancies were noticed: the same bounding box appeared multiple times as an internal node.

As such, the implementation was modified in a way inspired by the work presented in [8]. The underlying tree structure was changed to be a dictionary (or hash map), in which the key-value pairs represented a node and a list of its children. Instead of setting a node's left and right child, the list of children is extended. This implementation removed redundancy from internal nodes, reducing the height of the tree and preventing repeated queries against the same bounding box. Figure 3.2 shows a scenario in which a bounding box appears multiple times in the tree. The bounding box for triangle b completely contained the box for triangle d, and as such the bottom-up method for tree construction created a parent node for both box B and box D that was identical to box B. This redundancy initially expressed itself as a bug where nodes were deleted from the tree as it was being created, causing inconsistencies and incorrect query results.

This tree representation lends itself to a clean implementation. Algorithm 1 shows the pseudocode for generating the BVH. The pseudocode (using some Python notation) assumes the following:

- `root` is the root of the tree object being generated.
- `tree` is the internal representation of the tree and is a hash map, in which the key is an AABB, and the value is a list of either AABBs or triangular faces.

- The AABB constructor takes a list of faces or boxes and generates an AABB containing all objects passed to it.
- `faces` is a set (only unique faces).

Algorithm 1 VRCC-3D+ Tree Creation: Bottom Up.

```
function MAKE TREE(faces)
    if faces.length=1 then
        face ← faces.pop()
        root ← AABB(face)
        tree[root] ← face
        return root
    end if
    (left, right) ← PARTITION(faces)
    lbox ←MAKE TREE(left)
    rbox ←MAKE TREE(right)
    root ← AABB(lbox, rbox)
    tree[root].append(lbox)
    tree[root].append(rbox)
    return root
end function
```

4. DUAL GRAPH PARTITIONING (DGP)

In a three-dimensional object represented by the ANSI boundary representation, every face contains three vertices connected by three edges. This is directly analogous to a graph $G = (V, E)$, in which the set of vertices in the graph is the vertices of the triangular faces in 3-space, and an edge in the graph is a connection between two vertices in the facial boundary representation of the object. The dual graph G^* can be constructed as a graph in which each face is represented by a vertex, and an edge represents two faces that share two vertices in V; connectivity in G^* guarantees that two faces are spatially close.

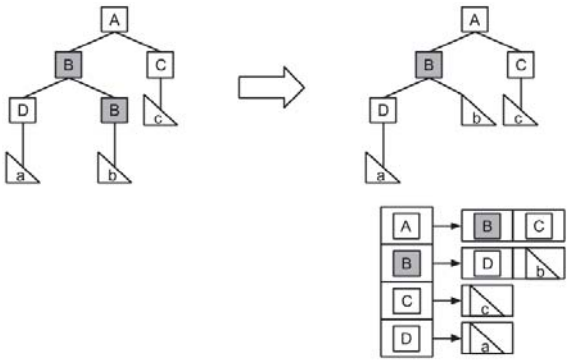


Figure 3.2: Redundancy in a BVH. In a bottom up approach, AABB B is a box that contains face b, but also fully encloses bounding box D. As such, it appeared multiple times in the tree, introducing redundancy and unnecessary calculations.

The dual graph can be used to partition the faces into groups of faces that are spatially close to each other. By choosing an arbitrary starting face in a set of faces, a breadth-first search to create a spanning tree that contains half of the vertices in the graph ensures that all faces in the partition are connected in the dual graph, and as such are spatially close relative to each other.

It is known in this implementation that the dual graph will be a sparse graph; every vertex has exactly three neighbors because every vertex represents a triangular face. As a sparse graph, it is efficient to represent it as an adjacency list. Algorithm 2 shows how the dual graph partitioning generates two partitions of faces.

5. EXPERIMENTAL DESIGN

The purpose of this experiment is to determine the effect that the dual graph partitioning scheme has on the runtime and quality of the tree. An algorithm in which *partition()* was implemented to split the set of faces in half by index was used as a control. It was then modified to accept and use the adjacency list of the dual graph to determine how the faces should be split when recursively creating the tree.

Algorithm 2 Dual Graph Partitioning.

```
function DUAL PARTITION(faces, adj)
    right ← set()
    currFace ← faces.pop()
    nextF ← queue()
    right ← {currFace}
    goal ← faces.length/2
    while right.length < goal do
        adjFaces ← faces ∩ adj[currFace]
        adjFaces ← adjFaces − right
        numToAdd ← goal − right.length
        if numToAdd > 3 then
            numToAdd ← 3
        end if
        toPush ← set(adjFaces[0 : numToAdd])
        right ← right ∪ toPush
        faces ← faces − toPush
        ENQUEUE(nextF, toPush)
        currFace ←DEQUEUE(nextF)
    end while
    return faces, right
end function
```

This implementation has some interesting features that evolved as the VRCC-3D+ implementation was growing, specifically that the bounding hierarchy is represented as a dictionary/hash map, and each internal node has a set that contains the children. This has an interesting side-effect: while the tree is constructed as a binary tree, if two grouped faces have the same bounding box, then the internal node in the tree will expand and have more than two children. This directly affects the size and average height of the tree, making those aspects of the tree smaller.

All timing was performed on an Intel Core i5 processor, running at 3.2GHz, with 16GB of RAM. All implementations are written in Python, using the *timeit* module to collect runtime information. Timing does not include the creation of the adjacency list unless otherwise specified; this is a one time cost that can be amortized over every tree creation.

5.1. CONFIGURATION 1

For the purposes of testing VRCC-3D+, a collection of 68 .obj files was created, each file depicting one of the VRCC-3D+ relations between two objects. Each of these files portrays two polyhedrons ranging from tetrahedrons (four triangular faces) to spheres (with 2000 triangular faces per polyhedron). These files were used for testing each of the BVH creation implementations.

The BVH creation time was averaged over 100 executions on each of the 68 sample files using both partitioning schemes. The number of internal nodes and average box volume was collected when using each partitioning scheme for every file.

5.2. CONFIGURATION 2

For a BVH, the act of determining which primitives a geometry can intersect is called querying the tree. Determining which faces from an object could intersect with faces from another object is called a box-box query (because boxes from one BVH are queried against boxes in the second).

Every .obj file contains two discrete objects. The box-box query time between the two objects was averaged over 100 runs for the same file set as in Configuration 1 for both partitioning schemes by implementing the algorithm presented in Algorithm 3.

5.3. CONFIGURATION 3

The need to create the adjacency list is an additional overhead that could impact performance. However, because the faces of the objects should never change their position relative to each other, the adjacency list can be precalculated and stored . This has a known memory cost (linear in the number of faces in the object). If

Algorithm 3 Box-Box Query.

```
function Box Box Query(tree1, node1, tree2, node2)
    isect ← dict()                          ▷ Dictionary of (node, list of nodes) pairs
    if Intersect(node1, node2) then
        for n1 ∈ tree1[node1] do
            for n2 ∈ tree2[node2] do
                if isAABB(n1) and isAABB(n2) then
                    isect ← isect+Box Box Query(tree1, n1, tree2, n2)
                else if isAABB(n1) then
                    isect ← isect+Box Box Query(tree1, n1, tree2, node2)
                else if isAABB(n2) then
                    isect ← isect+Box Box Query(tree1, node1, tree2, n2)
                else
                    isect[node1].append(node2)
                end if
            end for
        end for
    end if
    return isect
end function
```

memory is a problem, this list could be generated every time the BVH is generated. In this configuration, the time to create the adjacency list is averaged over 100 runs on the same set of files as in Configuration 1.

5.4. CONFIGURATION 4

To test scalability, each algorithm was executed 100 times on a file with two objects with significantly larger numbers of faces (9144 and 14624). The same statistics as in Configuration 1 were collected.

6. RESULTS

All times in these results are reported in seconds. All volume measurements are presented as cubed world units. Box and whisker plots are used to report quartile values over the input file set.

6.1. CONFIGURATION 1

Table 6.1 shows that, on average, the time to create a tree using DGP is about 15% longer than using a naïve partitioning scheme. Figure 6.1 shows that for smaller objects, in the best case, DGP can run as quickly as a naïve method.

The average box volume is also shown in Table 6.1 and Figure 6.3; note that the volume is, on average 5% smaller when the tree is generated using DGP. Table 6.1 and Figure 6.2 reflect the size of the tree through the number of internal nodes. A BVH created using DGP is, on average, slightly smaller than one created with a naïve partitioning method. In some cases, an increase is seen, but the average and median volumes are smaller when using DGP.

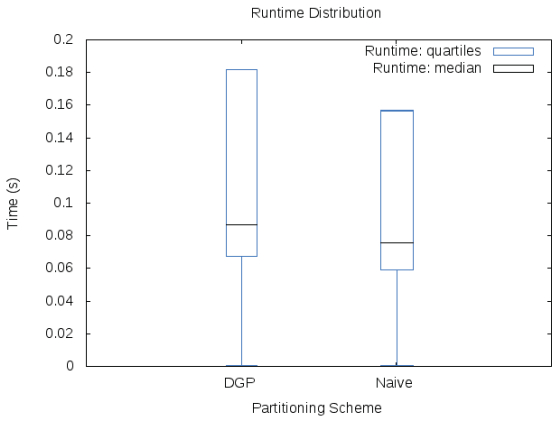


Figure 6.1: Range of Runtime for DGP and Naïve Partitioning.

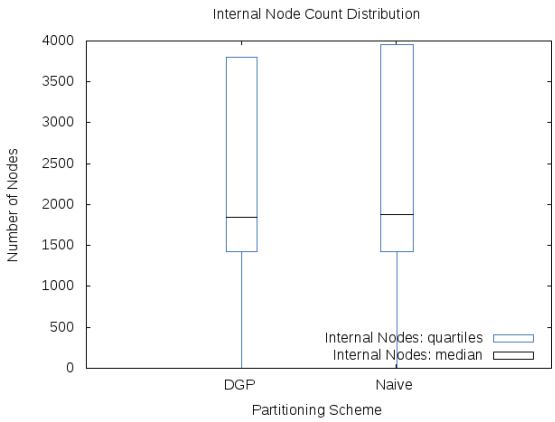


Figure 6.2: Range of Number of Nodes for DGP and Naïve Partitioning.

Table 6.1: Average and Standard Deviation: Configuration 1.

	μ: runtime	σ: runtime	μ: box volume	σ: box volume
DGP	0.105168340627	0.0714088494841	1.8130619326	3.24021971564
Naïve	0.090840193454	0.0613155721599	1.96238023672	3.16972935732

	μ: node count	σ: node count
DGP	2202.60294118	1491.67271031
Naïve	2275.45588235	1556.73862926

6.2. CONFIGURATION 2

Table 6.2 shows the average execution time of a box-box query between two objects. Again, DGP shows an improvement over the naïve partitioning process.

6.3. CONFIGURATION 3

Table 6.3 shows that the overhead of creating the list is small; it is less than half of the time required to generate the tree. If memory is a limitation, the adjacency list can be generated at tree creation, though the runtime will be half again as long.

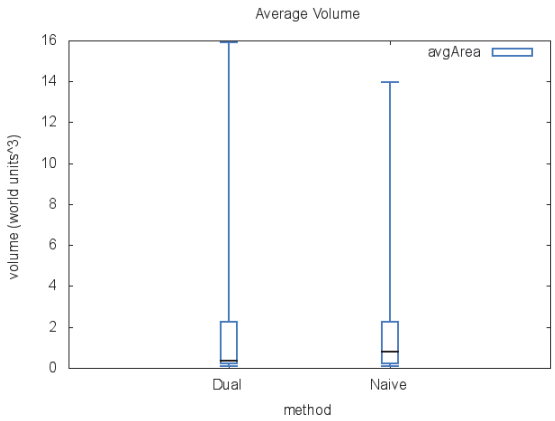


Figure 6.3: Range of Box Volume for DGP and Naïve Partitioning.

Table 6.2: Average and Standard Deviation of Query Runtime Over Test Files.

	μ: runtime	σ: runtime
DGP	0.528159871522	0.574440745841
Naïve	0.641162978411	0.51794431845

Table 6.3: Average and Standard Deviation of Adjacency List Creation Over Test Files.

Average	Standard Deviation
0.0472587228873	0.0422061185621

6.4. CONFIGURATION 4

The average creation time, number of nodes, and box volume taken from trees created on significantly larger objects are shown in Table 6.4. Note the mixed results; for one object, DGP generates fewer boxes but has a higher box volume, while for the other object, DGP results in more boxes with a smaller average box volume.

Table 6.4: Creation Time, Tree Size, and Box Volume on Larger Objects.

	Faces	Average Creation Time	Average Number of Nodes
DGP	9144	0.8753760600090027	15768
Naïve	9144	0.7410803198814392	15712
DGP	14642	1.4204304003715515	25688
Naïve	14642	1.193346061706543	25863

	Average Box Volume
DGP	271249.03166667151
Naïve	290342.60701300012
DGP	285103.56916561484
Naïve	265531.88658817636

7. CONCLUSIONS

7.1. CONFIGURATIONS 1 AND 2

The results from the first and second testing configurations are encouraging for the DGP scheme. Using DGP introduces a slight increase in the average BVH creation time (approximately 0.014 seconds, a 15% increase on the tested input object files). However, the query time to determine whether two objects could possibly intersect by performing a box-box query decreases by more than .11 seconds; this is almost an 18% improvement in runtime. The gain in query performance is almost eight times the slowdown introduced in the BVH creation.

To put this in perspective, consider a scene with n objects, requiring exactly n AABB creations. In the worst case, the VRCC-3D+ implementation would perform $\binom{n}{2}$ box-box queries ($O(n^2)$) to exactly determine the three-dimensional spatial relationships. Determining the obscuration relation requires additional queries to the tree; one of the predicates used to determine the obscuration is whether an object is in front of another.

In determining the obscuration between two objects, the implementation of VRCC-3D+ projects the three dimensional objects onto the view plane. If the two projections overlap, a number of evenly distributed points is chosen from the overlap. A ray starting at the camera and passing through each of these points is queried against the BVH to determine which object is passed through first. The precision and correctness of this operation increases as the number of points distributed across the overlapping projection increases. If p points are chosen from the intersection of the projection, then the number of ray queries made against the tree is $O(p)$.

It can be seen that the number of queries made to the tree quickly surpasses the number of tree creations: $O(n) < O(n^2) + O(p)$. As such, the time savings are significant. Not even the time overhead to create the adjacency list on the fly is enough to negate the benefit of using DGP over a naïve partitioning mechanism.

The box volume tells an interesting story. The average and median box volumes are smaller when using DGP. However, on occasion, DGP produces a larger total bounding volume than the naïve partitioning. While using the spanning tree of the dual graph ensures that the bounding volumes are spatially close, the starting point used in generating the tree can have an effect on the overall shape of the enclosing bounding volumes.

7.2. CONFIGURATION 3

As shown in the testing results from Configuration 3, the time it takes to generate the adjacency list is smaller than the time of tree generation. In cases where there is not enough memory to create and store the adjacency list for all objects on initialization, the list may be generated as the tree is being built. The additional runtime does not cause the increase in the creation time to overshadow the benefits to the query time. Indeed, the total time to create the adjacency list and generate the tree is smaller than the time savings in a single query.

7.3. CONFIGURATION 4

The results from testing Configuration 4 show that the DGP scales as well as a naïve partitioning scheme. The increase in runtime remains a constant factor ($\sim$ 18%). The trees generated show some inconsistencies; for one of the objects, the tree created using DGP has more nodes than the one created with the naïve partitioning method, but has a significantly smaller average box volume. For the other, larger object, this relationship is flipped: DGP results in a larger average box volume, but a smaller number of nodes. This suggests that for any partitioning, the shape of the object affects the results. Other work has been done in which the primitives are partitioned along the longest axis of the overall bounding box [5]; DGP would perhaps benefit here from a different starting point for the breadth-first search.

As a field, computer science is about problem solving with an emphasis on efficiency and elegance. A given solution may not work well on all representations of a problem; multiple approaches to a solution are beneficial as it allows flexibility in the application of the solution to different problem representations. Computer graphics and multimedia are very broad subjects: objects can be represented in a myriad of ways, making a single approach to analysis and computation tools nearly impossible. Bounding Volume Hierarchies are powerful tools; they allow for efficient collision queries by exploiting the logarithmic complexity of tree structures to quickly prune expensive calculations. Dual Graph Partitioning is an approach to generating these tree structures that can be applied to representations of scenes that do not lend themselves well to other tree creation algorithms. It is another tool in the large toolbox of powerful multimedia analysis mechanisms available to the modern computer scientist.

8. FUTURE WORK

Improving the quality and efficiency of the BVH implementation in VRCC-3D+ is an ongoing process. Optimizing the improvements gained by introducing DGP and the hash-map based tree representation will continue with the integration of other bottom-up tree improvements, such as those suggested in [5].

Areas for improvement include analyzing the effect that different breadth-first search starting points has on the quality of the resulting BVH and determining the feasibility and practicality of using threading to build pieces of the tree in parallel.

9. ACKNOWLEDGMENTS

Thanks to the students in the CS 390 Undergraduate Research course at Missouri S&T: Spencer Cramm, Emerson Lentz, and Emilio Navarro.

References

[1] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org, visited 2014-10-27.

[2] Timo Aila, Tero Karras, and Samuli Laine. On Quality Metrics of Bounding Volume Hierarchies. In *Proceedings of the 5th High-Performance Graphics Conference*, pages 101–107. ACM, 2013.

[3] Julia Albath, Jennifer L. Leopold, Chaman L. Sabharwal, and Anne M. Maglia. RCC-3D: Qualitative Spatial Reasoning in 3D. In *Proceedings of the 23rd International Conference on Computer Applications in Industry and Engineering*, pages 74–79, November 2010.

[4] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. In *23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 171–180, 1996.

[5] Yan Gu, He Yong, Jayvon Fatahalian, and Guy Blelloch. Efficient BVH Construction via Approximate Agglomerative Clustering. In *Proceedings of the Fifth High Performance Graphics 2013 (HPG '13)*, pages 81–88, 2013.

[6] Jörg Haber, Marc Stamminger, and H-P Seidel. Enhanced Automatic Creation of Multi-purpose Object Hierarchies. In *Proceedings of the Eighth Pacific Conference on Computer Graphics and Applications, 2000.*, pages 52–437. IEEE, 2000.

[7] Elizabeth G Houghton, Robert F Emnett, James D Factor, and Chaman L Sabharwal. Implementation of a Divide-and-Conquer Method for Intersection of Parametric Surfaces. *Computer Aided Geometric Design*, 2(1):173–183, 1985.

[8] T. Kim, B. Moon, D. Kim, and S. Yoon. Random-Accessible Compressed Bounding Volume Hierarchies. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):273–286, 2010.

[9] Thomas Larsson. *Adaptive Bounding Volume Hierarchies for Efficient Collision Queries*. PhD thesis, Mälardalen University, 2009.

[10] Christian Lauterbach, Michael Garland, Shubhabrata Sengupta, David Luebke, and Dinesh Manocha. Fast BVH construction on GPUs. *Computer Graphics Forum*, 28(2):375–384, 2009.

[11] J David MacDonald and Kellogg S Booth. Heuristics for Ray Tracing Using Space Subdivision. *The Visual Computer*, 6(3):153–166, 1990.

[12] C. L. Sabharwal and J. L. Leopold. Smooth Transition Neighborhood Graphs For 3D Spatial Relations. In *IEEE Symposium Series Workshop on Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP)*, pages 8–15, 2013.

[13] Chaman L. Sabharwal, Jennifer L. Leopold, and Nathan Eloe. A More Expressive 3D Region Connection Calculus. In *Proceedings of the 2011 International Workshop on Visual Languages and Computing (in conjunction with the 17th International Conference an Distributed Multimedia Sysems (DMS 11))*, pages 307–311, August 2011.

[14] Ingo Wald, Solomon Boulos, and Peter Shirley. Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies. *ACM Transactions on Graphics*, 26(1), 2007.

[15] Ingo Wald, William R Mark, Johannes Günther, Solomon Boulos, Thiago Ize, Warren Hunt, Steven G Parker, and Peter Shirley. State of the Art in Ray Tracing Animated Scenes. In *Eurographics 2007 State of the Art Reports*, 2007.

[16] Bruce Walter, Kavita Bala, Milind Kulkarni, and Keshav Pingali. Fast Agglomerative Clustering for Rendering. In *IEEE Symposium on Interactive Ray Tracing, 2008 (RT 2008).*, pages 81–86. IEEE, 2008.

[17] Douglas B. West. *Introduction to Graph Theory.* Prentice Hall, 2nd edition, 2001.

V. A More Efficient Representation of Obscuration for VRCC-3D+ Relations

Nathan Eloe *Chaman L. Sabharwal*

nwe5g8@mst.edu *chaman@mst.edu*

Jennifer L. Leopold

leopoldj@mst.edu

Missouri University of Science and Technology

ABSTRACT

VRCC-3D+ is an implementation of a region connection calculus that qualitatively determines the spatial relation between two 3D objects in terms of connectivity and obscuration. The eight connectivity relations are conceptually the same as RCC8, but calculated in 3D rather than 2D. The fifteen obscuration relations are calculated using the projection of the 3D objects on a particular 2D plane and the distance of the objects from the viewpoint. Herein we present a smaller, more precise set of VRCC-3D+ obscuration relations that retains the qualities of being jointly exhaustive and pairwise disjoint. However, this new set of relations overcomes two problems that existed in the previous set of fifteen relations: (1) lack of a precise mathematical definition for a key predicate, *InFront*, and (2) lack of an intuitive mapping of converse relations.

1. INTRODUCTION

Qualitative spatial reasoning (QSR) in two dimensions is a well-studied field, and includes models such as the connectivity-based RCC systems [3, 10, 12], and obscuration-based systems such as LOS-14 [6], OCS-14 [7], and OCC [8]. These systems, while expressive, do not accurately portray the real world wherein objects exist and are perceived in three dimensions, not two. As computing power increases and the need to analyze three-dimensional data (e.g., stereoscopic video, robotic vision, etc.) increases, two-dimensional reasoning systems can be inefficient, or even inadequate, for sophisticated applications.

To ameliorate the shortcomings of two-dimensional QSR systems, Albath et al. developed RCC-3D [2], which eventually evolved into VRCC-3D+ [13]. VRCC-3D+ uses composite relations that express both connectivity and obscuration from a given perspective. The connectivity-based relations are the RCC8 relations (DC, EC, EQ, PO, TPP, TPPc, NTPP, NTPPc) defined in three dimensions; these relations have been an ongoing focus of optimization and refinement in the implementation as a QSR system [5]. The obscuration portion of the composite relations are refinements on the basic concepts of no obscuration (nObs), partial obscuration (pObs), equal obscuration (eObs), and complete obscuration (cObs). Over time these relations have been enhanced to improve their expressive power.

There are three criteria that the relations must meet to maintain the quality of the QSR system: the set of relations must be Jointly Exhaustive and Pairwise Disjoint (JEPD), every relation should map to exactly one converse relation, and the relations should have an intuitive mapping to natural language. If the relations are not jointly exhaustive, there will be physical configurations of objects that simply cannot be expressed by any relation. Relations that are not pairwise disjoint will

result in ambiguous classification of object configurations. An intuitive mapping of the relations to natural language aids in human usefulness and usability of the system, and ensures that the expressive power of the system does not become needlessly complex; relations that cannot be differentiated in natural language typically do not add to the reasoning power of the system and overburden the computational complexity. Herein the authors focus on refining the obscuration terms of the composite VRCC-3D+ relations.

2. BACKGROUND AND RELATED WORK

2.1. OCCLUSION

Occlusion of one object by another object is contextually dependent on the observer's location (usually called the view point, the perspective reference point, or the center of perspective projection) relative to the objects. It follows that the occlusion decision can be made from the projection on a view plane. QSR applications are interested in deriving spatial obscuration relations and classification from projection of 3D objects on a 2D projection plane.

There are two types of projections as shown in Figure 2.1: parallel and perspective. Both have their advantages and disadvantages. The parallel projection is easier to compute, but loses the concept of depth. With the perspective projection, the object is scaled by the distance from the view point then projected; depth information is preserved. Obscuration predicates are based on two parameters: the perspective projection in a plane and depth (distance of the object from the perspective reference point).

The terms *in front*, *occulusion*, and *closer* are closely related. In natural language, the term *in front* between two objects A and B is synonymously interpreted as "A is in front of B", "A occludes B", and "A is closer than B".

(a) Parallel Projection. (b) Perspective Projection.

Figure 2.1: The Differences Between Parallel and Perspective Projections. In both cases, object A obscures object B.

2.2. QSR AND OCCLUSION

One of the best known obscuration-based QSR systems is LOS-14 [6] which was introduced by A.P. Galton in 1994. It classifies regions based on what can be seen in the Lines Of Sight (LOS) from a given perspective. Fourteen relationships are defined based on obscuration (or the lack thereof) from a given viewpoint.

Another occlusion-based calculus is ROC-20 [11]. It is similar to LOS-14, but extends it to add support for concave objects, which allows for mutual obscuration. Every spatial relationship in ROC-20 is defined in terms of the occlusion that is present and an RCC8 relationship. This system is significantly more expressive than LOS-14 and can apply to a greater number of cases, as it correctly handles concave regions.

The Occlusion Calculus (OCC) was introduced by Kohler in 2002 [8] and characterizes relationships between objects by their respective projections into an image plane. The author states that the information obtained is from one perspective, and as such, this system should be paired with other QSR methods to get a fuller picture. The system sacrifices expressiveness for reduced computational and reasoning complexity.

Guha et al. introduced OCS-14 in 2011 [7]. This model was designed to correct for insufficiencies in earlier occlusion models that made them infeasible for use in computer vision. Earlier methods had not accounted for whether the occluder was a moving object or part of the static background, and whether or not the visible part of an object was a connected blob or a fragment. As OCS-14 is designed for computer vision, feasibility of computation is a concern, but not expressive power.

2.3. RCC-3D

RCC-3D [2] was designed by Albath et al. to consider three dimensions, be computationally feasible, and give the most comprehensive spatial information about the system possible. Initially designed for use in analyzing the evolution of skeletal structures and other physical attributes, RCC-3D used the concepts of connectivity and obscuration to accomplish the design goals of completeness and computational feasibility. Because RCC-3D was to be used in visualizing physical changes over time, a GUI was deemed necessary. The resulting implementation was named VRCC-3D [1]. However, conceptual ambiguities that were uncovered in the implementation resulted in an evolution of the system, resulting in a revised model called VRCC-3D+ [13].

2.4. VRCC-3D+

Initially the obscuration portion of the VRCC-3D+ relationships simply were determined by overlapping boundaries and interiors of the projections of the objects in an image plane; the relations were limited to no obscuration (nObs), partial obscuration (pObs), complete obscuration (cObs), and equal obscuration (eObs). As the implementation of the system progressed, it became clear that a vital piece of information was missing; there was no concept of which object was obscuring the

other. As such, an additional ternary predicate was added called *InFront*. For two objects A and B, possible values for $InFront(A, B)$ were: YES (A is in front of B), NO (B is in front of A), and E (A and B are equidistant).

The ternary *InFront* predicate was used to refine the concepts of nObs, pObs, eObs, and cObs to express whether an object obscured the other, whether an object was obscured by the other (thereby adding _c to the relation name), or whether they obscured each other (thereby adding _e to the relation name). Some of these relations had an ambiguous combination of predicate values. As such, some of the relations were split, expanding the total number of relations to 15, as shown in Table 2.1.

Table 2.1: The 15 Current VRCC-3D+ Obscuration Relations.

	IntInt	**IntExt**	**ExtInt**	**InFront**
nObs	F	T	T	YES
nObs_c	F	T	T	NO
nObs_e	F	T	T	EQUAL
pObs1	T	T	T	YES
pObs2	T	F	T	YES
pObs_c1	T	T	T	NO
pObs_c2	T	T	F	NO
pObs_e	T	T	T	EQUAL
cObs	T	T	F	YES
cObs_c	T	F	T	NO
cObs_e1	T	T	F	EQUAL
cObs_e2	T	F	T	EQUAL
eObs_e	T	F	F	EQUAL
eObs_c	T	F	F	NO
eObs	T	F	F	YES

3. IMPROVEMENTS TO THE RELATIONS

The first step in improving the obscuration relations is dealing with the mathematically imprecise predicate *InFront*. The ternary nature of this predicate and lack of rigorous mathematical definition led to different interpretations of the same scene by different entities. To replace this predicate, two new predicates are proposed: *Obscures* ($o(A, B)$) and *ObscuredBy* ($o_c(A, B)$). The *Obscures* predicate is defined as follows:

Let $f_O(x, y)$ be a function that maps the point (x, y) on the image plane back to the point (x', y', z') in object O that projects to the point (x, y) and is closest to the image plane. If no point in object O projects to point (x, y), then $f_O(x, y) = (\infty, \infty, \infty)$. Also, let C be the location of the camera in world coordinates. The *Obscures* predicate for objects A and B is defined in Equation 1.

$$o(A, B) = \begin{cases} T & : \ \exists\, x, y \ s.t. |C - f_A(x, y)| < |C - f_B(x, y)| < \infty \\ F & : \ otherwise \end{cases} \tag{1}$$

In natural language, the meaning of this predicate is that it evaluates to true if there is a point at which the projections overlap and, within that projection, the first object hides some part of the second object. The definition of the converse relation $o_c(A, B)$ is simply $o_c(A, B) = o(B, A)$.

Note that the *Obscures* predicate only considers points at which the projection overlaps. This ameliorates cases such as that shown in Figure 3.1. If we remove the condition that the distance between the camera and each of the two points be finite, object B would be reported to obscure object A at a point where the object A does not have a projection in the image plane.

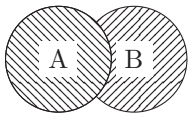


Figure 3.1: Partial Obscuration: Object A Obscures Object B.

Table 3.1: Mapping of *InFront* to o, o_c.

InFront	$o(A, B)$	$o_c(A, B)$
YES	T	F
NO	F	T
EQUAL	F(or T)	F(or T)

Table 3.1 shows the mapping of the original *InFront* predicate values to the values of the new *Obscures* and *ObscuredBy* predicates. Note that a value of EQUAL for the *InFront* predicate will map to either both o and o_c being true (T), or both being false (F).

The table of obscuration relations is rewritten as shown in Table 3.2. The first simplification of the relation set follows directly from the predicate extension: when there is no obscuration between two objects, the projections do not overlap. As such o and o_c will always be false. It follows that nObs and nObs_c are impossible relations; only nObs_e is allowed by the predicate set, and only when the values of the new predicates are both false. This leads to the obscuration characterizations in Table 3.3.

3.1. HANDLING PATHOLOGICAL CASES

Consider the projection shown in Figure 3.2a. In this image, object B is partially obscuring object A, and A is partially obscuring object B. Under the previous set of 15 obscuration relations, the only way to express this would be cObs_e, when

Table 3.2: The 15 Current VRCC-3D+ Obscuration Relations with o and o_c.

	IntInt	**IntExt**	**ExtInt**	o	o_c
nObs	F	T	T	T	F
nObs_c	F	T	T	F	T
nObs_e	F	T	T	F/T	F/T
pObs1	T	T	T	T	F
pObs2	T	F	T	T	F
pObs_c1	T	T	T	F	T
pObs_c2	T	T	F	F	T
pObs_e	T	T	T	F/T	F/T
cObs	T	T	F	T	F
cObs_c	T	F	T	F	T
cObs_e1	T	T	F	F/T	F/T
cObs_e2	T	F	T	F/T	F/T
eObs_e	T	F	F	F/T	F/T
eObs_c	T	F	F	F	T
eObs	T	F	F	T	F

Table 3.3: The Reduced Set of VRCC-3D+ Obscurations (nObs and nObs_c removed).

	IntInt	**IntExt**	**ExtInt**	o	o_c
nObs_e	F	T	T	F	F
pObs1	T	T	T	T	F
pObs2	T	F	T	T	F
pObs_c1	T	T	T	F	T
pObs_c2	T	T	F	F	T
pObs_e	T	T	T	F/T	F/T
cObs	T	T	F	T	F
cObs_c	T	F	T	F	T
cObs_e1	T	T	F	F/T	F/T
cObs_e2	T	F	T	F/T	F/T
eObs_e	T	F	F	F/T	F/T
eObs_c	T	F	F	F	T
eObs	T	F	F	T	F

pObs_e is more intuitively correct. The reason for this is that the interior of the projection of object A does not intersect with the exterior of the projection of object B, resulting in an (intuitively incorrect) identification of cObs as the base obscuration type.

(a) (b)

Figure 3.2: Two Examples of Mutual Obscuration. (a) A's projection completely contained in B's projection, (b)A's projection partially overlaps B's projection.

To address this issue, the structure of the relations themselves are examined. Currently, relations have a base of either nObs, pObs, eObs, or cObs. Appended to this base are the refinements of converse (_c) and equality (_e). To simplify this definition, a consistent structure is proposed: obscurations will have the form xObs$y$, where x correlates to the *extent* of obscuration (Table 3.4) and y corresponds to refinements on the obscuration (Table 3.5). To clarify the meaning of mutual obscuration, a new suffix is introduced: _m.

A Cartesian product of the prefixes and suffixes show that there are 16 possible obscuration relations. However, it has already been shown that there can only be a single version of nObs; there is not a way to map suffixes directly to prefixes. As such, each relation must be individually handled.

3.1.1. Partial Obscuration (pObs). For partial obscuration, all cases where both objects are visible must be considered. The definitions of pObs and pObs_c remain unchanged. The characterization of pObs_e does not change, and has values of false for both o and o_c.The mutual refinement for pObs must handle the

Table 3.4: Prefix and Extent of Obscuration.

Prefix (x)	**Meaning**
n	No Obscuration
p	Partial Obscuration
e	Equal Obscuration
c	Complete Obscuration

Table 3.5: Suffix and Obscuration Refinement, with Mapping to o and o_c.

Sufffix (y)	**Meaning**	$o(A, B)$	$o_c(A, B)$
[**none**]	Obscures	T	F
_c	Is Obscured By	F	T
_e	Equally Obscure Each Other	F	F
_m (new)	Mutually Obscure Each Other	T	T

case shown in Figure 3.2a as well as that shown in Figure 3.2b. The case shown in Figure 3.2b is straightforward. Figure 3.2a is more complicated: it must handle when object A is either object. As such, it maps to two characterizations of pObs_m. Table 3.6a shows the new characterizations for all pObs relations.

3.1.2. Equal Obscuration (eObs). Equal obscuration, by definition, occurs when the size and shape of the projections are identical; the values of the IntInt, IntExt, and ExtInt predicates will always be T, F, and F, respectively. The eObs_e obscuration should only occur if two objects are identical. Mutual equal obscuration can occur (Figure 3.3), so that case must be handled. Table 3.6b shows the characterizations of the new eObs relations.

Figure 3.3: Equal Mutual Obscuration.

3.1.3. Complete Obscuration (cObs). By definition, complete obscuration means that one object cannot be seen. As such, there is no cObs_m relation. Table 3.6c shows the cObs characterizations.

Table 3.6: The new VRCC-3D+ Obscuration Characterizations.

(a) New pObs characterizations.

	IntInt	**IntExt**	**ExtInt**	o	o_c
pObs	T	T F	T	T	F
pObs_c	T	T	T F	F	T
pObs_e	T	T	T	F	F
pObs_m	T	T F T	F T T	T	T

(b) New eObs characterizations.

	IntInt	**IntExt**	**ExtInt**	o	o_c
eObs	T	F	F	T	F
eObs_c	T	F	F	F	T
eObs_e	T	F	F	F	F
eObs_m	T	F	F	T	T

(c) New cObs characterizations.

	IntInt	**IntExt**	**ExtInt**	o	o_c
cObs	T	T	F	T	F
cObs_c	T	F	T	F	T
cObs_e	T	T F	F T	F	F

3.2. IDENTIFICATION OF CONVERSE OBSCURATIONS

One of the problems with the old set of VRCC-3D+ obscuration relations was that there were cases where there was no consistent intuitive mapping from a relation to its converse. For example, nObs_e, pObs_e, and eObs_e map to themselves as

converses, which is logical; if A and B obscure each other, then B and A should also obscure each other. However, there was not a single cObs_e relation but two. These two relations were the converse of each other. This inconsistency hindered both the implementation and reasoning with the system; it muddled the meaning of the _e suffix. Under this new relation set, every normal (no suffix) obscuration's converse relation is the converse obscuration (named with the _c suffix). The mutual (_m) and equal (_e) relations map to themselves as converse. Table 3.7 shows the full set of obscurations and their identified converse relation.

Table 3.7: Full Obscuration Relation Set with Identified Converse Relations.

	IntInt	**IntExt**	**ExtInt**	o	o_c	**Converse**
nObs_e	F	T	T	F	F	nObs_e
pObs	T	T F	T	T	F	pObs_c
pObs_c	T	T	T F	F	T	pObs
pObs_e	T	T	T	F	F	pObs_e
pObs_m	T	T F T	F T T	T	T	pObs_m
eObs	T	F	F	T	F	eObs_c
eObs_c	T	F	F	F	T	eObs_c
eObs_e	T	F	F	F	F	**eObs_e**
eObs_m	T	F	F	T	T	eObs_m
cObs	T	T	F	T	F	cObs_c
cObs_c	T	F	T	F	T	cObs
cObs_e	T	T F	F T	F	F	cObs_e

4. EFFECT ON VRCC-3D+

The new set of obscurations directly benefits the implementation of VRCC-3D+ on two fronts: it is easier to verify that the implementation of the predicates is correct (leading to more correct results), and the reduced set of obscurations and direct mapping to a converse relation improves the computational complexity of relation determination. Migration from using the ternary *InFront* to two binary predicates immediately improved the unit test pass rate for obscuration relations from 29% to 77%. The expected output for the unit tests was determined by visual inspection; moving to the more mathematically precise predicates caused the implementation to more closely emulate human perception; several of the common errors reported stem from the perspective point being different for the person analyzing the file and the implementation, and more are due to floating point rounding errors. The majority of the errors that were fixed stemmed from an incorrect suffix on the base obscuration. Also, the use of two binary predicates instead of a single ternary predicate makes it trivial to implement a decision tree predicate picker similar to that presented in [4], which has been shown to improve the speed of computation.

The primary improvement in computational complexity is due to the unique mapping of a relation to a converse relation. In order to fully describe a scene containing objects A and B using VRCC-3D+, the system must compute the relationship between objects A and B, but also between B and A (the converse relation). To calculate an RCC8 relation in three dimensions, the computational complexity is in the worst case $O(f_a \times f_b)$, where f_a and f_b are the number of faces in objects A and B, respectively. Generating the projections for objects A and B is $O(f_a)$ and $O(f_b)$, respectively; calculating the values of the predicate values is dependent on the (non-constant) complexity of performing intersection operations on the projections

and the cost of ray casting to determine which object is closer to the view point. Regardless of the implementation of these operations, the cost of computing the values of the predicates is more than linear. In contrast to this, if the relationship between objects A and B is known, and both parts of the composite relation have well-defined converses, the relationship between objects B and A can be determined to be the converse of each part of the composite relation: a lookup operation with complexity $O(1)$.

Every QSR system is designed as a balance of three criteria: ease of reasoning with the system, computational complexity, and expressive power of the system. An improvement to one aspect of the system comes at the cost of another. RCC-3D (the system that over time became VRCC-3D+) was initially designed to balance the three: by using compound relations, high expressive power and low computational complexity could be obtained without sacrificing too much in the way of ease of reasoning with the system. Herein, by reducing the 15 obscuration relations to 12 and introducing the concept of mutual obscuration as a refinement, both computational complexity and ease of reasoning have been improved.

Table 4.1 shows the new set of 34 composite relations present in VRCC-3D+ (a reduction from the 46 relations stemming from the old set of 15 obscuration relations), and also serves to illustrate the importance of working to minimize the number of obscuration relations; if the four obscuration relations with multiple characterizations were expanded to 9 separate obscurations, the number of VRCC-3D+ relations would grow to 50 relations. Usage of a QSR system becomes increasingly more complex as the set of relations in the system increases. An illustrative example of this is the composition table for the system which increases the speed of classifying relations by reducing the number of possible relations between two objects; in a scene with three objects A, B, and C, if the relationships between A and B, and B and C are known, the composition table reduces the set of relationships that are possible

between objects A and C. This can be used in conjunction with a decision tree [4] to speed up computation. Calculation of this table has a non-constant polynomial complexity in the number of relationships.

Table 4.1: Mapping of RCC8 Relations to Obscuration Relations.

	nObs_e	pobs	pObs_c	pObs_e	pObs_m	eObs	eObs_c	eObs_e	eObs_m	cObs	cObs_c	cObs_e
DC	X	X	X		X	X	X			X	X	
EC	X	X	X		X	X	X			X	X	
PO		X	X	X	X	X	X		X	X	X	
EQ								X				
TPP							X				X	X
TPPc						X				X		X
NTPP											X	
NTPPc										X		

5. FUTURE WORK

Future work will focus on mapping the revised VRCC-3D+ relations to natural-language terms suitable for end-user applications involving spatial querying. Preliminary efforts in this direction have commenced for the VRCC-3D+ connectivity relations [9]. Given the ambiguity of natural-language terms such as *in front/behind*, *occludes*, and *closer/nearer*, it may prove difficult to find unambiguous mappings for the mathematically precise VRCC-3D+ obscuration relations. Extensive human experiment studies will need to be conducted, and likely domain-specific ontologies will have to be developed for the relation-to-term associations.

6. CONCLUSIONS

In this paper, simplifications to the 15 obscuration relations present in VRCC-3D+ have been analyzed and presented. This change in the mathematical set of relations improved the computational correctness from 27% to 77%. The VRCC-3D+ obscuration relations are now easier to understand and computationally easier to implement because of the introduction of a new predicate for classification and a new class of obscuration.

References

[1] Julia Albath, Jennifer L. Leopold, and Chaman L. Sabharwal. Visualization of Spatio-Temporal Reasoning Over 3D Images. In *Proceedings of the 2010 International Workshop on Visual Languages and Computing (in conjunction with the 16th International Conference on Distributed Multimedia Systems)*, DMS '10, pages 277–282, 2010.

[2] Julia Albath, Jennifer L. Leopold, Chaman L. Sabharwal, and Anne M. Maglia. RCC-3D: Qualitative Spatial Reasoning in 3D. In *Proceedings of the 23rd International Conference on Computer Applications in Industry and Engineering*, pages 74–79, November 2010.

[3] B. Bennett. Spatial Reasoning with Propositional Logics‘. In J Doyle, E Sandewall, and P Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, KR '94, San Francisco, CA., 1994. Morgan Kaufmann.

[4] Nathan Eloe, Jennifer L. Leopold, Chaman L. Sabharwal, and Douglas McGeehan. Efficient Determination of Spatial Relations Using Composition Tables and Decision Trees. In *Proceedings of the IEEE Symposium on Computational Intelligence for Multimedia, Signal, and Vision Processing*, CIMSIVP '13, pages 1–7, 2013.

[5] Nathan Eloe, Jennifer L. Leopold, Chaman L. Sabharwal, and Zhaozheng Yin. Efficient Computation of Object Boundary Intersection and Error Tolerance in VRCC-3D+. In *Distributed Multimedia Systems '12*, 2012.

[6] A.P. Galton. Lines Of Sight. In *AISB Workshop on Spatial and Spatio-Temporal Reasoning*, 1994.

[7] Prithwijit Guha, Amitabha Mukerjee, and K. S. Venkatesh. OCS-14 : You Can Get Occluded in Fourteen Ways. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, IJCAI '11, pages 1665–1670, 2011.

[8] Christian Kohler. The Occlusion Calculus. In *Cognitive Vision Workshop*, ICVW '02, 2002.

[9] Jennifer L. Leopold, Chaman L. Sabharwal, and Katrina Ward. Spatial Relations Between 3D Objects: The Association Between Natural Language, Topology, and Metrics. In *Proceedings of the 2014 International Workshop on Visual Languages and Computing*, VLC '14, 2014. (To appear).

[10] Jihong Ouyang, Qian Fu, and Dayou Liu. A Model for Representing Topological Relations Between Simple Concave Regions. In *Proceedings of the 7th international conference on Computational Science, Part I: ICCS 2007*, ICCS '07, pages 160–167, Berlin, Heidelberg, 2007. Springer-Verlag.

[11] David Randell, Mark Witkowski, and Murray Shanahan. From Images to Bodies: Modelling and Exploiting Spatial Occlusion and Motion Parallax. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, volume 1 of *IJCAI'01*, pages 57–63, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[12] David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions and Connection. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Proceedings of the 3rd Conference on Principles of Knowledge Representation and Reasoning*, KR '92, pages 165–176. Morgan Kaufmann, San Mateo, California, 1992.

[13] Chaman L. Sabharwal, Jennifer L. Leopold, and Nathan Eloe. A More Expressive 3D Region Connection Calculus. In *Proceedings of the 2011 International Workshop on Visual Languages and Computing (in conjunction with the 17th International Conference an Distributed Multimedia Sysems (DMS 11))*, pages 307–311, August 2011.

SECTION

3. CONCLUSIONS

Qualitative Spatial Reasoning (QSR) is a powerful tool that has the potential to revolutionize the way computers interact with the world. Unfortunately, most QSR systems have severe limitations, such as being limited to two dimensions or being purely theoretical. RCC-3D was a first attempt to be an implemented system that was computationally feasible, worked in three dimensions, and enabled temporal reasoning. However, it still exhibited some shortcomings, including a lack of a visual user interface. VRCC-3D, and later VRCC-3D+, ameliorated many of the weaknesses present in the original system, RCC-3D. Over time, VRCC-3D+ has grown into a powerful theoretical system with an accompanying implementation.

This collection of work has provided a series of improvements to VRCC-3D+ that have increased the ease and correctness of implementation, made reasoning with the system simpler while retaining its expressive power, and have demonstrated a potential application domain for a completed implementation. Improvements in tolerance handling have allowed implementations to produce results that agree with human perception, an important aspect of such a system. Optimizations to the computational complexity and speed of calculations have led to an implementation more suitable for real-time reasoning. As a whole, this body of work has brought QSR closer to enabling the computer to accurately and quickly reason about the world.

BIBLIOGRAPHY

[1] ATI Stream Technology. http://www.amd.com/stream, visited 2012-04-27.

[2] Chronos Group: OpenCL. http://www.khronos.org/opencl/, visited 2012-04-27.

[3] NVIDIA Technologies: CUDA. http://developer.nvidia.com/category/zone/cuda-zone, visited 2012-04-27.

[4] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org, visited 2014-10-27.

[5] Timo Aila, Tero Karras, and Samuli Laine. On Quality Metrics of Bounding Volume Hierarchies. In *Proceedings of the 5th High-Performance Graphics Conference*, pages 101–107. ACM, 2013.

[6] Julia Albath, Jennifer Leopold, Chaman Sabharwal, and Kenneth Perry. Efficient Reasoning with RCC-3D. In *The 4th International Conference on Knowledge Science, Engineering, and Management (KSEM 2010)*, pages 470–481. KSEM, September 2010.

[7] Julia Albath, Jennifer L. Leopold, and Chaman L. Sabharwal. Visualization of Spatio-Temporal Reasoning Over 3D Images. In *Proceedings of the 2010 International Workshop on Visual Languages and Computing (in conjunction with the 16th International Conference on Distributed Multimedia Systems)*, DMS '10, pages 277–282, 2010.

[8] Julia Albath, Jennifer L. Leopold, Chaman L. Sabharwal, and Anne M. Maglia. RCC-3D: Qualitative Spatial Reasoning in 3D. In *Proceedings of the 23rd International Conference on Computer Applications in Industry and Engineering*, pages 74–79, November 2010.

[9] Renee Baillargeon. Representing the Existence and the Location of Hidden Objects: Object Permanence in 6- and 8-month-old Infants. *Cognition*, 23(1):21–41, 1986.

[10] Stephen T. Barnard. Disparity Analysis of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):333–340, July 1980.

[11] B. Bennett. Spatial Reasoning with Propositional Logics‘. In J Doyle, E Sandewall, and P Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, KR ’94, San Francisco, CA., 1994. Morgan Kaufmann.

[12] Brandon Bennett, Anthony G. Cohn, Paolo Torrini, and Shyamanta M. Hazarika. A Foundation for Region-based Qualitative Geometry. In Werner Horn, editor, *ECAI*, pages 204–208. IOS Press, 2000.

[13] Blender Foundation. blender.org - Home. `http://blender.org`, April 2013.

[14] Eliseo Clementini, Jayant Sharma, and Max J. Egenhofer. Modelling topological spatial relations: Strategies for query processing. *Computers and Graphics*, 18(6):815 – 822, 1994.

[15] A G Cohn, B Bennett, J Gooday, and N Gotts. RCC: a Calculus for Region-Based Qualitative Spatial Reasoning. *GeoInformatica*, 1:275–316, 1997.

[16] A G Cohn, B Bennett, J Gooday, and N Gotts. Representing and Reasoning with Qualitative Spatial Relations about Regions. In O Stock, editor, *Temporal and spatial reasoning*. Kluwer, 1997.

[17] Anthony G. Cohn, Brandom Bennett, John Gooday, and Micholas Mark Gotts. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *Geoinformatica*, 1(3):275–316, October 1997.

[18] D. Demirdjian and T. Darrell. Motion Estimation from Disaprity Images. In *Proceedings of the Eighth IEEE International Conference on Computer Vision ICCV*, volume 1, pages 213–218, 2001.

[19] Tiansi Dong. A Comment On RCC: from RCC to RCC++. *Journal of Philosophical Logic*, 37:319–352, 2008.

[20] David Dunning and Emily Balcetis. Wishful Seeing: How Preferences Shape Visual Perception. *Current Directions in Psychological Science*, 22(1):33–37, February 2013.

[21] Max J. Egenhofer. Deriving the Composition of Binary Topological Relations. *J. Vis. Lang. Comput.*, 5(2):133–149, 1994.

[22] Nathan Eloe, Jennifer L. Leopold, and Chaman L Sabharwal. Spatial Temporal Reasoning Using Image Processing, Physics, and Qualitative Spatial Reasoning. *International Journal of Software Engineering and Knowledge Engineering*. In Review.

[23] Nathan Eloe, Jennifer L. Leopold, Chaman L. Sabharwal, and Douglas McGeehan. Efficient Determination of Spatial Relations Using Composition Tables and Decision Trees. In *Proceedings of the IEEE Symposium on Computational Intelligence for Multimedia, Signal, and Vision Processing*, CIMSIVP '13, pages 1–7, 2013.

[24] Nathan Eloe, Jennifer L. Leopold, Chaman L. Sabharwal, and Zhaozheng Yin. Efficient Computation of Object Boundary Intersection and Error Tolerance in VRCC-3D+. In *Distributed Multimedia Systems '12*, 2012.

[25] Nathan Eloe, Chaman L Sabharwal, and Jennifer L. Leopold. A More Efficient Representation of Obscuration for VRCC-3D+. *Polibits*. In Review.

[26] Nathan Eloe, Joseph A Steurer, Jennifer L. Leopold, and Chaman L Sabharwal. Dual Graph Partitioning for Bottom-Up BVH Construction. *Journal of Visual Languages and Computing*, 2014.

[27] A.P. Galton. Lines Of Sight. In *AISB Workshop on Spatial and Spatio-Temporal Reasoning*, 1994.

[28] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. In *23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 171–180, 1996.

[29] Yan Gu, He Yong, Jayvon Fatahalian, and Guy Blelloch. Efficient BVH Construction via Approximate Agglomerative Clustering. In *Proceedings of the Fifth High Performance Graphics 2013 (HPG '13)*, pages 81–88, 2013.

[30] Prithwijit Guha, Amitabha Mukerjee, and K. S. Venkatesh. OCS-14 : You Can Get Occluded in Fourteen Ways. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, IJCAI '11, pages 1665–1670, 2011.

[31] Jörg Haber, Marc Stamminger, and H-P Seidel. Enhanced Automatic Creation of Multi-purpose Object Hierarchies. In *Proceedings of the Eighth Pacific Conference on Computer Graphics and Applications, 2000.*, pages 52–437. IEEE, 2000.

[32] Elizabeth G Houghton, Robert F Emnett, James D Factor, and Chaman L Sabharwal. Implementation of a Divide-and-Conquer Method for Intersection of Parametric Surfaces. *Computer Aided Geometric Design*, 2(1):173–183, 1985.

[33] A. Karlsson. GIS and Spatial Extensions with MySQL. *MySQL Developer Zone, http://dev. mysql. com/tech-resources/articles/4.1/giswith-mysql. html*, 2007.

[34] T. Kim, B. Moon, D. Kim, and S. Yoon. Random-Accessible Compressed Bounding Volume Hierarchies. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):273–286, 2010.

[35] David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2010.

[36] Christian Kohler. The Occlusion Calculus. In *Cognitive Vision Workshop*, ICVW '02, 2002.

[37] Yohei Kurata. A Strategy for Drawing a Conceptual Neighborhood Diagram Schematically. In *Proceedings of the 5th international conference on Diagrammatic Representation and Inference*, Diagrams '08, pages 388–390, Berlin, Heidelberg, 2008. Springer-Verlag.

[38] Thomas Larsson. *Adaptive Bounding Volume Hierarchies for Efficient Collision Queries*. PhD thesis, Mälardalen University, 2009.

[39] Christian Lauterbach, Michael Garland, Shubhabrata Sengupta, David Luebke, and Dinesh Manocha. Fast BVH construction on GPUs. *Computer Graphics Forum*, 28(2):375–384, 2009.

[40] Jennifer L. Leopold, Chaman L. Sabharwal, and Katrina Ward. Spatial Relations Between 3D Objects: The Association Between Natural Language, Topology, and Metrics. In *Proceedings of the 2014 International Workshop on Visual Languages and Computing*, VLC '14, 2014. (To appear).

[41] Canlin Li, Jiajie Lu, Chao Yin, and Lizhuang Ma. Qualitative Spatial Representation and Reasoning in 3D Space. In *Proceedings of the 2009 Second International Conference on Intelligent Computation Technology and Automation - Volume 01*, ICICTA '09, pages 653–657, Washington, DC, USA, 2009. IEEE Computer Society.

[42] J David MacDonald and Kellogg S Booth. Heuristics for Ray Tracing Using Space Subdivision. *The Visual Computer*, 6(3):153–166, 1990.

[43] Adaleigh Martin, Ian Kottman, Robyn Littleton, Jennifer Leopold, Chaman L. Sabharwal, and Nathan Eloe. Determining Minimal Transitions Between VRCC-3D+ Relations. In *Distributed Multimedia Systems '12*, 2012.

[44] Koji Mineshima, Mitsuhiro Okada, and Ryo Takemura. Two Types of Diagrammatic Inference Systems: Natural Deduction Style and Resolution Style. In *Proceedings of the 6th international conference on Diagrammatic representation and inference*, Diagrams'10, pages 99–114, Berlin, Heidelberg, 2010. Springer-Verlag.

[45] Tomas Möller. A Fast Triangle-Triangle Intersection Test. *journal of graphics, gpu, and game tools*, 2(2):25–30, 1997.

[46] Karsten Mühlmann, Dennis Maier, Jürgen Hesser, and Reinhard Männer. Calculating Dense Disparity Maps from Color Stereo Images, an Efficient Implementation. *International Journal of Computer Vision*, 47(1–3):79–88, April 2002.

[47] A. Murta. A General Polygon Clipping Library. *Advanced Interfaces Group, Department of Computer Science, University of Manchester. On-line resource. URL: http://www. cs.-man. ac. uk/aig/staff/alan/software/gpc. html*, 2000.

[48] Jihong Ouyang, Qian Fu, and Dayou Liu. A Model for Representing Topological Relations Between Simple Concave Regions. In *Proceedings of the 7th international conference on Computational Science, Part I: ICCS 2007*, ICCS '07, pages 160–167, Berlin, Heidelberg, 2007. Springer-Verlag.

[49] J.R. Quinlan. Induction of Decision Trees. *Machine learning*, 1(1):81–106, 1986.

[50] J.R. Quinlan. *C4. 5: Programs for Machine Learning*, volume 1. Morgan kaufmann, 1993.

[51] David Randell, Mark Witkowski, and Murray Shanahan. From Images to Bodies: Modelling and Exploiting Spatial Occlusion and Motion Parallax. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, volume 1 of *IJCAI'01*, pages 57–63, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[52] David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions and Connection. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Proceedings of the 3rd Conference on Principles of Knowledge Representation and Reasoning*, KR '92, pages 165–176. Morgan Kaufmann, San Mateo, California, 1992.

[53] David A. Randell and Mark Witkowski. Using Occlusion Calculi to Interpret Digital Images. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *Proceedings of the 17th European Conference on Artificial Intelligence*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 432–436. IOS Press, August 2006.

[54] Rui M. P. Reis, Max J. Egenhofer, and João L. G. Matos. Conceptual Neighborhoods of Topological Relations Between Lines. In *SDH*, pages 557–574, 2008.

[55] Jochen Renz. *Qualitative Spatial Reasoning with Topological Information*. Springer-Verlag, 2002.

[56] Jochen Renz. Qualitative Spatial and Temporal Reasoning: Efficient Algorithms for Everyone. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 526–531, January 2007.

[57] Jochen Renz and Gérard Ligozat. Weak Composition for Qualitative Spatial and Temporal Reasoning. In Peter Beek, editor, *Principles and Practice of Constraint Programming – CP 2005*, volume 3709 of *Lecture Notes in Computer Science*, pages 534–548. 2005.

[58] Jochen Renz and Bernhard Nebel. Efficient Methods for Qualitative Spatial Reasoning. *Journal of Artificial Intelligence Research*, 2001.

[59] Pascal Rost, Lothar Hotz, and Stephanie von Riegen. Supporting Mobile Robot's Tasks through Qualitative Spatial Reasoning. In *Procceedings of the Ninth International Conference on Informatics in Control, Automation and Robotics (ICINCO 2012)*, volume 2, pages 394–399, 2012.

[60] Antonio J. Rueda Ruiz and Lidia M. Ortega. Geometric Algorithms on CUDA. In *GRAPP*, pages 107–112, 2008.

[61] C. Sabharwal. Survey of Implementations of Cross Intersection Between Triangular Surfaces. MDC Report Q0909 (Now Boeing at St. Louis, MO-USA), 1987.

[62] C. L. Sabharwal and J. L. Leopold. Smooth Transition Neighborhood Graphs For 3D Spatial Relations. In *IEEE Symposium Series Workshop on Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP)*, pages 8–15, 2013.

[63] Chaman Sabharwal and Jennifer L. Leopold. Reducing 9-Intersection to 4-Intersection for Identifying Relations in Region Connection Calculus. In *The 24th International Conference on Computer Applications in Industry and Engineering*, pages 118–123. CAINE, November 2011.

[64] Chaman Sabharwal and Jennifer L. Leopold. Smooth Transition Neighborhood Graphs for 3D Spatial Relations. In *2013 IEEE Symposium Series on Computational Intelligence*, 2013.

[65] Chaman L. Sabharwal, Jennifer L. Leopold, and Nathan Eloe. A More Expressive 3D Region Connection Calculus. In *Proceedings of the 2011 International Workshop on Visual Languages and Computing (in conjunction with the 17th International Conference an Distributed Multimedia Sysems (DMS 11))*, pages 307–311, August 2011.

[66] Sandro Santilli, Mark Leslie, Chris Hodgson, Paul Ramsey, Jeff Lounsbury, and Dave Blasby. PostGIS Manual. *Refractions Research Inc*, 2005.

[67] Maribel Yasmina Santos and Adriano Moreira. Conceptual Neighborhood Graphs for Topological Spatial Relations.

[68] Paulo E. Santos. Reasoning about Depth and Motion from an Observer's Viewpoint. *Spatial Cognition and Computation: An Interdisciplinary Journal*, 7(2):133–178, December 2007.

[69] M. Stocker and E. Sirin. PelletSpatial: A Hybrid Region Connection Calculus RCC-8 and RDF/OWL Reasoning and Query Engine. *Proceedings of Web Ontology Language (OWL): Experiences and Directions 2009 (OWLED 2009)*, 2009.

[70] Kazuko Takahashi. Reasoning about Relative Relationships in 3D Space for Object Extracted from Dynamic Image Data. In *Proceedings of the 2012 ECAI Workshop on Spatio–Temporal Dynamics*, pages 45–51, August 2012.

[71] James T. Todd. The Visual Perception of 3D Shape. *TRENDS in Cognitive Sciences*, 8(3):115–121, March 2004.

[72] Ingo Wald, Solomon Boulos, and Peter Shirley. Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies. *ACM Transactions on Graphics*, 26(1), 2007.

[73] Ingo Wald, William R Mark, Johannes Günther, Solomon Boulos, Thiago Ize, Warren Hunt, Steven G Parker, and Peter Shirley. State of the Art in Ray Tracing Animated Scenes. In *Eurographics 2007 State of the Art Reports*, 2007.

[74] Bruce Walter, Kavita Bala, Milind Kulkarni, and Keshav Pingali. Fast Agglomerative Clustering for Rendering. In *IEEE Symposium on Interactive Ray Tracing, 2008 (RT 2008).*, pages 81–86. IEEE, 2008.

[75] Douglas B. West. *Introduction to Graph Theory.* Prentice Hall, 2nd edition, 2001.

[76] D.W.S. Wong and J. Lee. *Statistical Analysis of Geographic Information With ArcView GIS and ArcGIS*, volume 1. Wiley, 2005.

[77] Jun Ye and Kien A. Hua. Exploiting Depth Camera for 3D Spatial Relationship Interpretation. In *Proceedings of the Multimedia Systems Conference*, February 2013.

VITA

Nathan Webster Eloe was born on in Dayton, OH. In May 2010, he received his B.S. in Computer Science and Physics from the Missouri University of Science and Technology, with minors in Mathematics and Theatre. Immediately following graduation, he began graduate studies at the same institution, and received his Ph.D. in Computer Science in May 2015. During his time as a student he worked for Lumate (previously IDC Projects) as a senior developer, working on implementing server communication software, database interfaces, and techologies using GIS information to provide geospatial information about a query.

He has published conference papers as a primary and secondary author; some of these papers are cited in this research.