基于三维模型的自动语义标注研究与应用

摘要

随着计算机辅助设计，三维虚拟化技术的快速发展，在它们的实际使用中产生了大量的三维场景和模型。针对它们的精确检索和分类因此成为有效利用这些数据的关键所在。语义标注技术在一定程度上解决了三维知识管理的难题，但在应用数据越来越趋向海量的今天，其本身也面临着新的挑战。这些挑战包括：如何缩短标注时间，减少标注成本，使其面对数据增长具有可扩展性；如何降低领域专家的依赖，使其趋向自动化；如何维护标注的一致性，抑制内容相互冲突的标注；以及如何在做到上述几点的同时，仍然保证标注内容与手动完成的相比同等的准确性。

本文以传统的语义标注技术为基础，进行针对标注半自动化过程的优化，来解决现有语义标注过程中重复劳动多，效率低下的问题。本文关注于网络三维文档，并根据其特点提出多侧面的本体模型，对三维场景建模中各领域通用的概念进行抽象以及复用；同时提出标注规则，帮助标注者抽象标注方法，以达到标注过程半自动化以及标注内容一致性可验证的目的。本文最后通过案例验证由文中提出方法所完成的标注和传统的手工标注比较起来，拥有同等的准确性，可以在后续的查询检索操作中起到有效的作用。

完成的工作有：  
一、面向网络三维场景，提出一套基于本体和规则的半自动化语义标注方法。给出完整的方法流程，实现框架以及应用场景。

二、提出针对三维场景文档的本体建模。给出用于描述三维场景建模基本概念及属性的通用本体的实现方式。在此基础上，描述从网络三维文档到本体实例的映射算法，该算法能够从三维文档中抽取物体的几何，空间关系等可视属性。

三、提出标注规则的语法和语义，能够灵活地表达特定的标注模式。在此基础上，完成标注引擎的实现算法，包括规则一致性检测，规则在文档中的匹配，以及规则的应用方法。

四、设计了面向网络三维场景基于本体和规则的半自动化语义标注原型系统，并在实际应用场景中验证标注过程半自动化的有效性和准确性

本文所提出的半自动化语义标注方法，基于传统使用本体的语义标注，继承了后者描述精确的特点。此外，通过对它进行改进，实现了利用规则进行标注逻辑抽象，从而进一步达到半自动化的目标。本文还验证了半自动生成的标注内容并没有比纯手动标注降低准确性。因此，该方法有效地提高了网络三维文档的标注效率，在大量数据的情况下，能避免标注过程中过多的重复工作，快速地应对新增的文件，是后续使用中全面的查询与检索操作的基础支撑，对三维场景的语义管理具有较高的参考价值。

关键词：三维模型，语义标注信息，WebGL, 自动语义标注，视频资源, 图像识别。

3D model-based Semantic Annotation Management Research and Application

ABSTRACT

In recent years, along with the rapid development of 3D and virtual interaction technology, 3D graphic computing has proven its important position in many major fields. As the 3D-related resources has been increased recently, 3D scene and object modeling have gained a lot of attractions and supports from various communities. Unfortunately software products that are capable of manipulating these works are still limited in management aspect, making it difficult to extend them to their fullest potential. Users nowadays demand higher requirements from the lower-layer manage components of a 3D software. First, the software products must be able to provide a powerful search engine, which can filter through a large amount of 3D objects to find the desired 3D models or scenes from provided information. Second, the structure, characteristic or type of a 3D object must be analyzed in a reasonable speed to support the traditional semantic search.

These requirements lead to a creative idea of annotating the semantic contents to 3D objects. Semantic annotation help users not only maintaining the internal data and structure, but also the semantic contents of a 3D object. These semantic contents include the color, characteristic, type and other information such as the usage for applications. To fulfill the feature of combining all the semantic contents together, the annotating process must meet two required conditions.

For the first condition, the method of annotation must have an acceptable accuracy. This means it must have the ability to provide correct and unique information of a 3D object in a specific field. This condition ensures that in the 3D software products, users can easily find their desired 3D objects without filtering too much redundant and unrelated results. Also, user can quickly get the same-type resultant models from a provided type of model.

Shanghai Jiao Tong University (SJTU) is a key university in China. SJTU was founded in 1896. It is one of the oldest universities in China. The University has nurtured large numbers of outstanding figures include JIANG Zemin, DING Guangen, QIAN Xuesen, Wu Wenjun, WANG An, etc.

The other condition, which is efficiency-related, becomes necessary when we have to deal with large amount of data. Assume that annotation itself is a complex, time-consuming and repeated process, the total time which includes duration of user's manual operation and annotation cannot keep up to the standard requirements of a search engine. High-efficient requirements comprise two things: first, there must be supported tools or computer software to let the annotation process become automatic or semi-automatic, and the semantic contents can have fast inquiry and access; second, the repeated process is able to be recognized to be executed automatically.

From the above conditions and requirements, we introduce a WebGL and 3D model-based automatic semantic annotation system. This system is capable of automatically acquire the semantic contents of an object through the data structure analysis or comparing the characteristics between models.

.

**Key words**: 3D Modelling，Semantic annotation，WebGL, Auto Semantic，Video Resources, Image Recognition.

目 录

[基于三维模型的自动语义标注研究与应用 I](#_Toc406619052)

[摘要 I](#_Toc406619053)

[ABSTRACT IV](#_Toc406619054)

[第一章 绪论 1](#_Toc406619055)

[1.1 研究背景与意义 1](#_Toc406619056)

[1.2 国内外研究现状 2](#_Toc406619057)

[1.2.1 WebGL的三维建模方法 2](#_Toc406619058)

[1.2.2 半自动的语义标注方法 3](#_Toc406619059)

[1.2.3 基于本体的自动语义标注方法 4](#_Toc406619060)

[1.2.4 针对图像识别的标注方法 5](#_Toc406619061)

[1.3 本文研究内容 6](#_Toc406619062)

[1.4.1 自动的语义标注方法研究概况 6](#_Toc406619063)

[1.4.2 三维模型图形比较方法的自动语义标注 7](#_Toc406619064)

[1.4 论文组织结构 8](#_Toc406619065)

[第二章 三维模型语义标注框架 9](#_Toc406619066)

[2.1 应用场景分析 9](#_Toc406619068)

[2.2 需求分析 11](#_Toc406619069)

[2.2.1 三维模型语义标注的用例分析 11](#_Toc406619070)

[2.2.2 视频语义标注的用例分析 12](#_Toc406619071)

[2.2.3 可靠性 13](#_Toc406619072)

[2.2.4 可用性 14](#_Toc406619073)

[2.2.5 方便性 14](#_Toc406619074)

[2.2.6 可扩展性 14](#_Toc406619075)

[2.3 面向WebGL三维模型的自动化语义标注总体框架 14](#_Toc406619076)

[2.4 平台目标及优势 15](#_Toc406619077)

[2.5 本章小结 15](#_Toc406619078)

[第三章 三维模型的自动语义标注 16](#_Toc406619079)

[3.1 背景 16](#_Toc406619081)

[3.2 图像分析及相似度比较方法 16](#_Toc406619082)

[3.3.1 二维图像的结构 16](#_Toc406619083)

[3.3.2 图像相似度比较 17](#_Toc406619084)

[3.3.3 图像转换Base64码存储 19](#_Toc406619085)

[3.3.4 工作流程 21](#_Toc406619086)

[3.3 图像的识别的语义标注方法 23](#_Toc406619087)

[3.4.1 三维模型转换二维图像 23](#_Toc406619090)

[3.4.2 二维图像的信息存储 23](#_Toc406619091)

[3.4.3 工作流程 24](#_Toc406619092)

[3.4 本章小结 25](#_Toc406619093)

[第四章 视频资源的语义标注 26](#_Toc406619094)

[4.1 背景 26](#_Toc406619096)

[4.2 基于Web的视频播放器 26](#_Toc406619097)

[4.2.1 设计目标 26](#_Toc406619098)

[4.2.2 模块划分 26](#_Toc406619099)

[4.2.3 Gstreamer多媒体框架 27](#_Toc406619100)

[4.2.4 工作模式 29](#_Toc406619101)

[4.3 视频的自动截取画面 31](#_Toc406619102)

[4.4 视频的语义标注 31](#_Toc406619103)

[4.5 本章小结 31](#_Toc406619104)

[第五章 系统实现与验证 32](#_Toc406619105)

[5.1 自动语义标注框架和总体架构 32](#_Toc406619106)

[5.2 三维模型自动语义标注的设计过程 33](#_Toc406619107)

[5.2.1 三维模型的导入和展示 33](#_Toc406619108)

[5.2.2 三维模型自动转二维图像方法实现 37](#_Toc406619109)

[5.2.3 自动识别图像的语义标注方法实现 43](#_Toc406619110)

[5.3 基于前段的图像基本分析及相似度比较 50](#_Toc406619111)

[5.3.1 前段的图像自动分析 50](#_Toc406619112)

[5.3.2 二维图像的相似度比较 51](#_Toc406619113)

[5.4 结果分析及讨论 57](#_Toc406619114)

[5.4.1 研究结果和意义 57](#_Toc406619115)

[5.4.2 该系统与其他语义标注系统的比较 59](#_Toc406619116)

[5.4.3 系统的不足之处 59](#_Toc406619117)

[5.5 本章小结 59](#_Toc406619118)

[第六章 总结与展望 60](#_Toc406619119)

[6.1 主要的研究工作总结 60](#_Toc406619121)

[6.2 未来的工作展望 61](#_Toc406619122)

[参考文献 62](#_Toc406619123)

[1 谢辞 66](#_Toc406619124)

[2 攻读学位期间发表的学术论文目录 67](#_Toc406619125)

# 绪论

## 研究背景与意义

近期几年来，伴随着三维技术和虚拟现实与计算机的发展 。3D计算机图形技术已经广泛的应用到各个领域，并且变得越来越重要。三维模型及场景收到计算机行业的欢迎和认可。三维文档的数量也因此而快速增长。然而如果没有成熟的软件技术进行支持。这些模型将在很多应用上无法管理, 从而限制了它们在应用中能起到作用。

基于以上原因, 针对三维模型或场景的应用对底层的管理软件提出了更高的需求。首先, 这些软件需要作为搜索引擎, 能够在大量的三维数据中检索满足指定条件的模型或者场景; 其次,它们必须能够迅速地抽取出这些模型的三维结构,来支持语义查询搜索,或者迅速的为三维模型找出特点以及分类。

这些需求的提出直接引发了对三维模型进行语义标注的想法。语义标注是指模型的提供者除了存储三维模型的数据和结构信息之外, 同时还要维护它们的语义。假如某个物体所属的类型,颜色，分类以及它在实际工程中用途等,并且将前后两者以某种方式关联起来。为了满足所提出的需求, 标注过程必须满足以下几个条件:

第一,标注知识的载体必须具有一定的准确性。它所代表的标注内容能够精确而无 二义性地表达一个领域知识。该条件是为了保证在后续的查询检索等应用中, 用户可以 准确地找到自己想要的三维模型, 同时不会看到太多与查询条件无关的场景，或者针对一种模型，可以推送类似该模型的同类。

第二,标注过程需要具有一定的高效性。这里论文研究的意义是如果针对海量数据是如何高效率的对其进行自动语义标注。若标注过程本身是一项繁琐而耗时，重复的工作, 则在它之上消耗的时间和人力最终将无法支撑其之后在检索上提供的优势和便利。

基于以上的需求和条件，本文提出一种基于三维模型WebGL的自动语义标注系统框架。这套系统利用分析模型本身的数据结构，或者进行比较三维模型之间的特性得到该模型的标注语义信息。

## 国内外研究现状

对于软件开发的质量和效率无法满足企业需求的问题，解决方法一是靠软件工程，厂商采用更科学、更规范的流程组织项目开发；二是靠软件技术。而就软件技术而言，平台化技术是软件产品发展的重要趋势。软件开发的问题是由多方面原因造成的，有市场的原因、有公司管理水平的原因，也有技术的原因。目前基于业务基础软件平台的开发方式是解决这些问题的一个比较好的解决办法。而模型驱动开发是有效实现业务与技术分离的方法，可以解决项目的业务需求复杂并且频繁变更的问题。

### WebGL的三维建模方法

WebGL是一种3D绘图标准，这种绘图技术标准允许把JavaScript和OpenGL ES 2.0结合在一起，通过增加OpenGL ES 2.0的一个JavaScript绑定，WebGL可以为HTML5 Canvas提供硬件3D加速渲染，这样Web开发人员就可以借助系统显卡来在浏览器里更流畅地展示3D场景和模型了，还能创建复杂的导航和数据视觉化。显然，WebGL技术标准免去了开发网页专用渲染插件的麻烦，可被用于创建具有复杂3D结构的网站页面，甚至可以用来设计3D网页游戏等等。

目前在国内外已经有很多大公司，游戏公司正在往WebGL这样先进的三维技术发展，不断的改善该技术。到目前为止，WebGL已经完美地解决了现有的Web交互式三维动画的两个问题：第一，它通过HTML脚本本身实现Web交互式三维动画的制作，无需任何浏览器插件支持;第二，它利用底层的图形硬件加速功能进行的图形渲染，是通过统一的、标准的、跨平台的OpenGL接口实现的。

WebGL标准已出现在Mozilla Firefox、Apple Safari及Google Chrome等流行浏览器中，这项技术支持Web开发人员借助系统显示芯片在浏览器中展示各种3D模型和场景。在2014年三月份，Epic游戏同Mozilla一同通过打造让人先睹为快的gEpic Soul 和 Swing Ninja demo在Firefox火狐浏览器中以接近原生的速度运行证明了Web在如何不断进化为一个强大的游戏平台。这个视频中首次展现的内容是虚幻4引擎无需插件运行在Web中。

Mozilla和Epic游戏此前通过将虚幻3引擎移植到Web中和展示Epic Citadel——使用Mozilla打造的JavaScript超级子集asm.js，展示了Web作为游戏平台的强大力量。在不到12个月的时间里，使用asm.js的优化将Web应用的性能从原生应用的40%提高到了67%，我们期待它能变得更快。这样WebGL现在带来的性能，远远比之前网页的三维展示技术提高了很多。性能越来越接近本地的应用程序，利用系统资源的效率也大大提高 。所有的主流浏览器都可以运行asm.js内容，但是目前只有Firefox提供了特别的优化，来确保最一致和平滑的体验。未来有望随着该技术的发展，主流的浏览器都可以支持该脚本，能够展示3D网页游戏及复杂3D结构的网站页面。

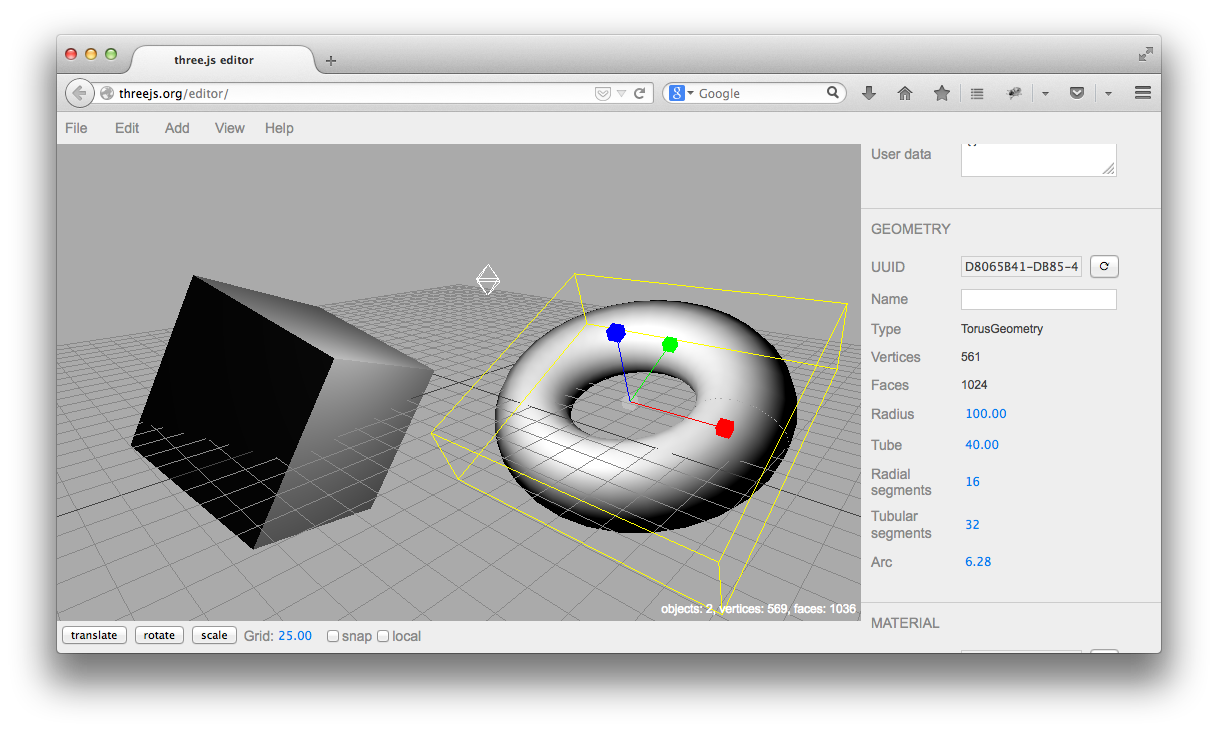


图1-1三维模型操作原型系统界面

Fig.1-1 3D Modelling Operation User Interface

### 半自动的语义标注方法

语义标注是为了实现语义以及其他基于语义信息的应用的基础。语义标记的目标是将本体实例数据与本体的类相互关联起来。手动的语义标注在现今的研究背景下已 经变得十分容易。然而，使用人工标注也常常伴随了许多问题的出现。因为人工操作可能是许多错误的来源，这其中包括 标注者对业务领域的不熟悉，标注者接受训练的程度不足，以及待标注文档的复杂度过 高以至于标注者无法胜任。另一方面，人工标注也因为成本太高而无法应用于海量数据 的标注工程中。自动或者半自动的标注过程随即成为解决这一问题的唯一途径，尤其是半自动化的标注方法，得到了大量的研究及引用，因为全自动对计算机的智能要求较高，在现阶段往往无法取得较好的效果。

关于自动化进行语义标注的方法存在三种，分别是监管条件下基于机器学习 的方法；无监管条件下基于机器学习的方法以及基于规则的标注方法。监管环境下基于机器学习的方法包括了两个阶段：标注阶段和训练阶段。在标注阶段中，输入数据是 结构化或者半结构化的文档，目标是识别这些文档中的实体以及它们之间的关系。在训练阶段中，实际的任务是学习将在后续标注过程中被使用到的标注模型。

在半自动的语义标注方法领域里面，D.Xu和H.Li两人通过了神经网络的方式来进行概念的学习。他们使用了Princeton Shape Benchmark(PSB) 数据库中的若干分类来进行手动标注，然后将后者送入神经网络的 分类器。通过这种方式，文档内的底层信息与高层语义知识之间的关系被建立了起来。 这种学习方式的缺点是比较耗时，并且需要一开始拥有大量的训练集数据。

同样在语义标注领域里面， R.Ohbuchi领导的研究提出了一种半监管环境下的学习方式，他们首先利用无监管的学习将一堆未标记的三维模型根据它们的特征分类到若干子空间中，随后在各自的小 范围内利用监管学习来进行更细致的分类。他们的实验结果显示这种做法比单纯监管或 无监管的方式要更加有效。

### 基于本体的自动语义标注方法

语义标注在最初应用的时候一般情况下都是使用关键字作为标注知识的载体。基于关键字的标注方式虽然因为自然语言的不精确性和随意性具有一定的歧义，在检索过程中准确率较低，但是从普通用户的角度来讲，却是十分适合的做法。例如在Facebook社交网络里面，用户上传图片的时候。系统后台会自动检测出照片文档里面所描述的信息进行提出关键字，或者允许用户对其照片通过圈信息方法进行图片的描述信息。从上述例子中可以看出，基于关键字的标注方式在现代的互联网环境下多用户的协同标注中具有广泛的应用。

在利用本体进行语义标注的研究中，L.Hollinkde的研究，他们建立了一套表示画作和画家等的本体，并利用它对某展览馆的作品进行了标注。他们建立了软件接口来支持用户针对画作名称，画作风格等信息的检索。该研究的缺点是，他们无法对画作中的一部分进行具体的标注，从而欠缺了图像的内容检索。作为该缺点的改进，J. Assfal 将特征抽取方法应用到体育比赛的视频中，来识别比赛中的关键事件，并对其进行自动标注。

同样在基于本体的自动语义标注领域里面，B. Popov等人总结了利用本体和自然语言处理在媒体新闻中抽取人事和企业信息的一般做法。S. Bloehdorn等的研究则采用了图像描述符的手段来匹配图像中的特殊区域，并对它们进行标注。

针对X3D三维模型文件的半自动化语义标注方法。同样使用本体的自动语义标注方法, Shi MengWei提出了一套基于本体面向网络三维文档的半自动语义标注框架，旨在保证标注正确性的前提下，利用计算机软件系统根据用户指定的规则针对三维场景文档进行自动标注。在该研究中，作者提出通用本体来抽象三维场景建模中的视觉信息，并提出算法来抽取该类信息。在此基础上构建标注规则和规则实施引擎来实现标注过程的半自动化。

### 针对图像识别的标注方法

基于图形图像识别的三维模型语义标注方法，最重要是能够建立一个三维模型与它相应所表示的语义信息之间的链接。这个链接之间，应该需要有一个桥梁。这个桥梁就是如何把网页上的三维模型数据结构转换为一个可识别的二维图像。经过图像识别的算法和相应的知识库能够得出其所表示的语义信息。

图像识别可能是以图像的主要特征为基础的。每个图像都有它的特征, 如字母A有个尖，P有个圈、而Y的中心有个锐角等。对图像识别时眼动的研究表明，视线总是集中在图像的主要特征上，也就是集中在图像轮廓曲度最大或轮廓方向突然改变的地方，这些地方的信息量最大。而且眼睛的扫描路线也总是依次从一个特征转到另一个特征上。由此可见，在图像识别过程中，知觉机制必须排除输入的多余信息,抽出关键的信息。同时, 在大脑里必定有一个负责整合信息的机制，它能把分阶段获得的信息整理成一个完整的知觉和映像。

到目前为止，研究关于图形图像的识别技术已经有很多人研究，并发表了许多文章关于该领域。在2010年，一个创业公司IQ Engine 开发了一款API，帮助网络零售商和应用开发者提供视觉搜索引擎。IQ Engine目前开发并维护着两个重要的图像识别API。其一名为Smart Camera。这一API帮助用户与产品和品牌标志互动。另一个API名为Smart Album，支持在网络相册和移动应用中进行照片分析和面部识别。后一个API可能正是雅虎需要的功能。IQ Engine开发的API已被许多知名公司使用。

综合上述所提到的目前三维模型的建模技术以及对三维模型的自动语义标注的研究。我们可以看到在该领域中已经有很多作者在研究并取得不少成果。近期几年来三维计算机图形技术已经广泛的应用到各个领域，并且变得越来越重要。尤其是针对三维模型或场景做语义标注的技术。目前对于语义标注的研究有很多种方法。其中有半自动语义标注方法，基于本体的语义标注方法，等等。 综合以上的研究现状以及已有的技术，本文研究的主要重点在于如何通过提取三维模型文档的视觉信息和图形比较算法来为大量的三维模型自动生成语义标注。

## 本文研究内容

### 自动的语义标注方法研究概况

综合上述研究现状及现在的研究热点，本论文将提出一套框架基于HTML5 WebGL, 能够实现针对三维模型或场景通过手势操作和针对三维模型的自动语义标注。能够保证标注内容正确性的前提下对标注过程进行效率的提升。主要的研究内容包括以下几点：

1. 提出一种算法来抽取并分析三维模型中的视觉和语义信息。这其中包括：分析并识别出三维文档中包含的视觉信息和特性。针对三维模型文档的信息特性来推测出同类的语义。
2. 通过两个三维模型或者整个三维场景的图像识别比较算法，计算出图像之间的相似度，从而得出模型之间的异同点，找出相似的地方。然后使用已有的语义标注信息，进行自动的三维模型自动语义标注。
3. 构建出原型系统。按照上述所说的思想和算法，搭建基于三维模型场景的语义标注的原型系统Leap Studio。该系统中包括三维模型文档的信息和特性提取。通过模型的比较特性（如颜色，材料，种类。。。）来对它进行自动语义标注。系统通过将新建的三维模型转换为canvas图像，与数据库中的样本模型进行图形比较找出异同点。从而为新模型自动生成语义标注。

### 三维模型图形比较方法的自动语义标注

三维模型的自动语义标注方法是该论文的重要技术，在该论文中提到两种方法可以进行3D模型的自动语义标注方法。方便能够帮助用户快速查找，分类自己所需要的三维模型。以及帮助用户在使用，浏览模型中系统自动推荐出类似的模型。让用户体验以及针对模型的操作效率提高。

本文提出一种标注方法是通过算法来抽取并分析三维模型中的视觉和语义信息。WebGL的三维模型本身就是json格式的文本，具有三维模型定义属性的数据结构和自带WebGL所定义的特殊数据结构。在这个json数据本身所包含着一个三维模型的形状，颜色，材料，坐标 等等信息。为了提高三维模型的语义标注工作的效率，本论文中将利用该信息设计出一种算法来获取这些模型的视觉信息。然后再针对每一种视觉信息提出将它们从三维文档中抽取出来的策略。以后面提取三维模型视觉信息的效率。

3D文档信息的获取

三维模型的特性比较

自动语义标注内容

3D 模型

可视化信息

模型特性异同信息

数据库

前端处理

后台处理

图1-2自动语义标注过程的流程图

Fig.1-2 Auto Semantic Annotation Flow Graph

以上的流程图为整个三维模型自动语义标注系统的流程图，从整个语义标注的过程中我们可以看到三维模型或场景将作为该系统的输入材料。将三维模型导入进来之后，通过系统的语义标注引擎由两个方式来获取到三维模型的语义信息；其中是上述所提到的三维模型可视化信息的提取方法。

## 论文组织结构

第一章“绪论”部分主要介绍本文的研究背景和国内外研究现状。

第二章“面向 X3D 的语义标注框架”通过应用场景分析介绍本文所提出的方法的主要框架和其涉及到的模块。并简单介绍标注结果的持久化形式。

第三章“三维场景文档的本体描述”提出本文用到的通用本体，文档本体和领域本体的概念。介绍了通用本体需要满足的条件，并提出一个通用本体的实现。最后提出从三维文档中抽取可视化信息的算法。

第四章“基于规则的语义标注研究”提出了本文用到的标注语言，标注规则的匹配和应用算法。

第五章“系统实现与验证”讨论了本文研究中所实现的原型系统的实现细节。并在此基础上用一个例子演示了完整的标注流程。

第六章“总结与展望”总结了本文的主要工作，并对未来可能的研究方向做出了展望。。第八章为结论，给本题目做总结并表达研究心得。

# 三维模型语义标注框架

本章将提出面向WebGL的自动化语义标注过程的总体框架。首先，我们将通过若干应用场景的分析来引导出标注框架的需求，并据此提出本文所研究的标注框架。随后，本章将介绍该框架内执行三维场景语义标注的一个典型流程。框架中的主要部件及它们所起的作用将在这之后得到简要的描述。



## 应用场景分析

近年随着三维文档的受欢迎程度不断提高，越来越多的应用中涉及到了针对三维场景的查询和检索。一个常见的应用场景例如一个使用工业零件三维模型库的制造业企业，它希望能根据特性或参数，精确地从大量的零件中找到匹配的模型，从而指导并完成其生产线上的加工，拼装等流程。该应用提出了这样的需求：首先，库中的每个模型都要携带额外的数据来描述它们的特性或参数；其次，这些信息必须能够精确地反应其所对应模型的领域信息。

另一个应用场景来源于一个家具装潢电子商务网站的用户。他们希望能够从该网站提供的大量居室场景中，根据其内部的某些细节特征，找到符合条件的房屋。这些查询请求可能包括房屋内地板的质量，某一件房间的采光条件，或者居室间的相互连接。该应用场景要求：场景文档所附加的语义信息不仅能够描述场景作为一个整体，还要能描述场景内部的组成部分及其相互关系。

为了满足这些应用需求，三维模型场景库的创建者或供应商需要对他们的三维文档进行语义标注工作。图 2-1 展示了语义标注的过程和它们的最终应用。在该图中，三维文档经过标注流程而产生了语义标注信息，这些信息被存储起来，并成为检索软件系统的知识库，为用户提供精确高效的查询服务。

对于标注工作来讲，它则需要满足以下几个要求：首先，标注所产生的信息需要以一种精确的方式来表达，从而保证用户的查询条件和他得到的结果都具有准确性；其次，三维模型场景中的基本视觉信息和物体间空间关系需要得到识别，并能够和它们在更高层上所表示的语义知识产生联系；最后，标注过程本身需要具有一定的高效性。对于大量的三维模型场景，标注人员将不需要为每一条标注内容进行手工输入。他们的部分工作需要由标注系统的自动化来代替。

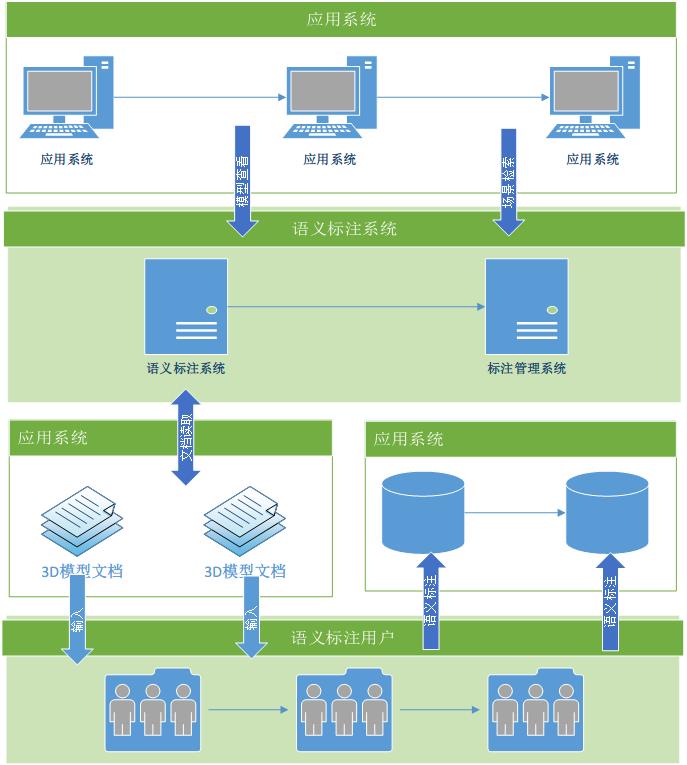


图2-3语义标注过程与其应用场景

Fig.2-3 Auto Semantic Annotation Progress and Application Scenario

## 需求分析

### 三维模型语义标注的用例分析

例如当我们需要将一个三维模型转换的二维图像与数据库的其他三维模型做比较的时候。假设三维模型数据库存放大量成千上万个模型数据。显然如果将一个三维模型与整个数据库里面的所有模型进行比较，为了同步他们之间的语义标注信息。这样的效率会特别慢，没有达到要解决海量数据的意义。

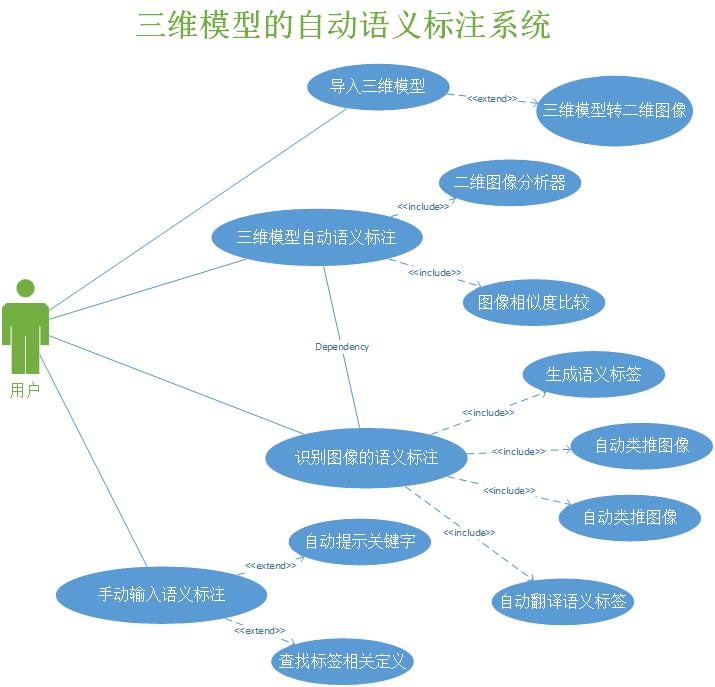


图3-1三维模型的自动语义标注用例图

Fig.3-1 3D model of automatic semantic annotation use case diagram

为了解决这样的问题，我们完全可以结合以上两种语义标注方法使用：

第一**：**首先要为导入的三维模型进行多视角截图，然后通过**方法1**从多个模型生成二维图像进行图像识别方法。并得到语义标注信息（见图方法1）。

第二**：**当三维模型拥有了语义标注信息之后。可对其进行分类，聚类模型操作。从**方法2** 可看到比较模型相似的自动语义标注方法的输入资源是从**方法1**图像识别自动语义标注方法得到的结果。之后通过访问数据库，在海量模型数据中找出类似同样的模型数据类型。得到更为相关性的模型数据。然后进行原三维模型与其他同类模型的相似度比较。再次进行语义标注，同步相似度比较高的模型之间的语义信息。

### 视频语义标注的用例分析

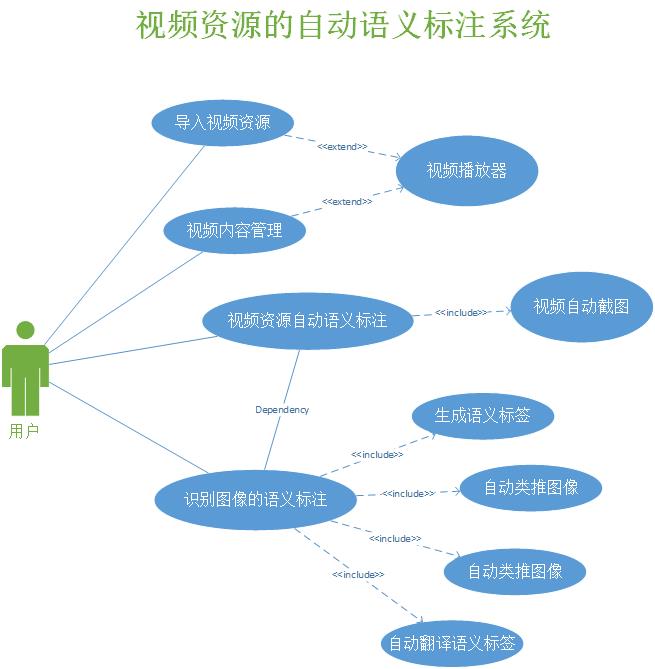


图4-2视频资源的自动语义标注用例图

Fig.4-2 Video of automatic semantic annotation use case diagram

### 可靠性

系统的可靠性体现在于系统能够准确的识别出本次输入签名的人是手机拥有者本人而是非他人的。该元素也是系统的核心技术和最有关键性的特征。根据目前的研究现状，系统能百分之百地识别出用户签名的习惯仍然未实现到。但可考虑使用某种识别机制使系统能尽量有最低的第一型及第二型错误的概率。

### 可用性

构造出的系统必须满足对隐私信息安全有不同要求的用户。因此系统需要有灵活性，能根据用户不同的需求来做出适当的认证。系统应允许用户能管理并随意更换签名的对比样本，并且还能够记录使用过程的信息，包括错误信息及提供汇报统计功能。同时与锁住手机模块相关的功能设置如锁住时间长度，允许错误的次数等也是很重要。从以上的需求分析，该认证系统的设置模块是不可缺少的部分。

### 方便性

若当认证的时候需要依赖于其他周围环境会对用户造成许多不方便的使用体验。如第三章所提到，使用手机像头来识别人脸或麦克风来做语音识别很多时候需要依赖外围环境的状态。因此这些认证方式不能保证在何时何地每次认证都能达到最高的效率。高的第一型及第二型错误会影响到认证系统的使用方便性和安全性。解决该问题的最好方案是减少系统对其他外围环境的依赖性。本题目的研究方向考虑把用户与设备间的互动只限制于通过触摸屏来进行验证。该互动方式和传统的Password/PIN/Pattern一样，但应用新的验证方式可仍然保证系统的安全性及效率性。

### 可扩展性

可扩展性也包括两方面：性能扩展和平台扩展。系统的开发过程中需要将每个功能模块划分清楚。各模块可独立操作且互相调用形成系统的完整运作流程。这样在后期工作可容易优化或添加新功能而不影响到整个系统的运作及结构。本系统属于基于安卓智能手机平台的认证系统，因此将系统扩展到其他移动设备平台是值得考虑的。使用跨平台的手机应用开发。。工具使开发系统的源代码可有重用性，从而减低开发其他平台同样应用的工作量。

## 面向WebGL三维模型的自动化语义标注总体框架

本文所提出的半自动化语义标注系统旨在不影响标注内容正确性的前下，将部分标注转化为可由计算机程序自动生成，从而减少手工重复工作。它的主要框架如图 2-2所示。其中，由灰色阴影标识的部分标识由计算机自动处理完成，其余则需要人工干预。该图由若干部件以及它们之间的连线组成，连线上方的数字表示在一个典型的标注流程中，各步骤的执行次序。

## 平台目标及优势

## 本章小结

本章通过实际的应用场景引出了本文所提出的面向基于WebGL三维模型的自动化语义标注系统的总体框架。本章随后利用一个典型的标注流程，简要描述了框架中各个模块的主要作用。最后，本章介绍了由标注框架产生的标注内容在持久化后的具体表现形式。后续章节将对这些模块进行详细展开，来讨论它们的内部实现细节。

# **三维模型的自动语义标注**

本章将提出面向WebGL的自动化语义标注过程的总体框架。首先，我们将介绍。



## 背景

近年随着三维文档的受欢迎程度不断提高，越来越多的应用中涉及到了针对三维场景的查询和检索。一个常见的应用场景例如一个使用工业零件三维模型库的制造业企业，它希望能根据特

## 图像分析及相似度比较方法

### 二维图像的结构

本文中的系统使用的图像分析以及比较方法对方法是以属于该图像的像素为操作对象。一个像素的结构是由不同的彩色值组成。颜色值可分成四种：R（红）、G（绿）、B（蓝）和A（亮度）。从程序设计的数据结构来看，可把每个像素看成一个数组有四个元素。从第0至第3个元素按序为R、G、B、A，四个元素的数值范围为0至255。各像素与属性之间的数据结构可从图4-4中看出。

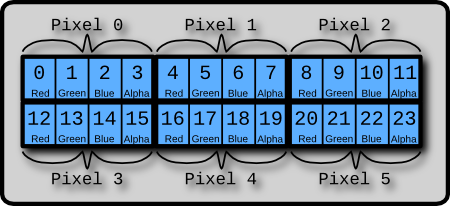


图3-3二维图像像素的数据结构

Fig.3-3 Two-dimensional image pixel data structure

从该论文中使用的图像分析以及比较算法。我们有一个像素跳转到下一个像素需要一个偏移量。该偏移量的计算方法为：

 （4-1）

其中offset为像素的偏移量，y为像素目前位置的纵坐标，w为像素所属的图片的宽度，x为目前位置的横坐标。

### 图像相似度比较

图像的对比流程是通过扫描两个目标图像的像素，再计算两者的差别率且导出差别的结果图案[31]。以比较两个像素结构的R、G、B、A值是否相等为方法，若两者的像素颜色对应则跳往下一个像素，否则会生成一个表示错误的像素，且差别率加一。当扫描工作结束后，签名对比模块将生成一幅差别图，由扫描过程中收集到的所有错误像素组成。另外从每笔画对获得的相似度计算出平均相似度。该相似度作为决定验证是否能通过的主要元素。若得出的平均相似度大于所设置的允许相似度，则验证通过，否则失败。



图3-4二维图像对比的差别结果图

Fig.3-4 the results of two-dimensional image comparison chart

所谓的二维图像相似度比较原始是从两个图像计算出的差别率。之所以数据的使用法被倒过来是因为针对该算法，三维模型所转换的二维图像的背景也考虑成一种颜色，而如果图像中三维模型的截图像素数量与图像的像素数量之差很大。因此得出的差别率会与两者之间的差别像素量反正比。

原始的差别率计算是通过取生成的差别像素与整幅图的像素之间的比率。公式如下：

 （4-2）

其中mismatch为差别率，ne为差别像素数量，w和h分别为图像对象的宽度和高度。

系统中计算出最后的三维模型所转换的二维图像的平均相似率公式如下：

 （4-3）

Average match为平均相似度，公式的分子部分是从1到n个属于同一类三维模型的图像转换的差别率总值，ns为图像的数量。

整体的三维模型转换二维图像集合比较过程可用以下的伪代码来形容。

Begin

　　//从参数获得二维图像的集合

　　设Compare Image [] = 三维模型转换的二维图像集合

　　设Origin Image = 数据库所获取到同一类的二维图像集合

　　　 //存放差别像素的数组

　　 设ErrorPixel = new Array

　　 //循环所有三维模型转换的二维图像集合进行比较

　　 For(i=0; i < Compare Image 数组长度; i++)

　　 {

　　 // 传入当前对象

　　 设imageA = 第i个需要比较图像；

　　 设imageB = Origin Image

　　 //扫描两个二维图像的像素

　　 For(j = 0; j < imageA.lastPixel; j++)

　　 {

　　 //若像素彩色相等则返回

　　 If(imageA .pixel[j] == imageB .pixel[j]) then return;

　　 Else //若彩色不相等

　　 {

　　 Mismatch++;

　　 //生成一个错误像素并进栈

　　 ErrorPixel.push(createErrorPixel);

　　 }

　　 }

　　 //计算第i个笔画的差别率

　　 设MisRate = MisMatch /(imageA.width \* imageB.height) \* 100;

　　 //从差别像素数组构造出差别结果图

　　 设DiffImage = createFrom(ErrorPixel);

　　 设TotalMatch = TotalMatch + MisRate;

　　 }

　　 //计算平均相似度

　　 FinalMatch = TotalMatch / A.length

　　 //从平均相似度是否符合要求给出相应的结果

　　 //DefaultMatch为允许通过匹配的相似率

　　 If (FinalMatch >= DefaultMatch)

//两张二维图片结果匹配

　 Return true

　　}

End

### 图像转换Base64码存储

Base64是一种以ASCII格式的字符形式来显示二进制码。Base64编码及译码技术在许多针对文字型数据处理的系统菜单应用相当广泛。本系统从签名输入区获得的签名信息也以Base64码表示。签名对象在与样本签名对比时也已Base64格式转发给签名比较模块。在存储签名样本的文件时原始的Base64码签名也需要通过Base64译码技术来转换成二进制才能存储成图片文件[31]。

Base64编码技术不仅有一种，但所有不同译码方法都以前62字符为共同的使用字符，包含A–Z、a–z和0–9。本系统的签名接受区导出的签名使用MIME方式的Base64译码，第63、64字符为“+”与“/”。表4-1是基于MIME的Base64码索引表[32]。

表3-1 Base64码索引表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Value | Char |  | Value | Char |  | Value | Char |  | Value | Char |
| 0 | A | 16 | Q | 32 | g | 48 | w |
| 1 | B | 17 | R | 33 | h | 49 | x |
| 2 | C | 18 | S | 34 | i | 50 | y |
| 3 | D | 19 | T | 35 | j | 51 | z |
| 4 | E | 20 | U | 36 | k | 52 | 0 |
| 5 | F | 21 | V | 37 | l | 53 | 1 |
| 6 | G | 22 | W | 38 | m | 54 | 2 |
| 7 | H | 23 | X | 39 | n | 55 | 3 |
| 8 | I | 24 | Y | 40 | o | 56 | 4 |
| 9 | J | 25 | Z | 41 | p | 57 | 5 |
| 10 | K | 26 | a | 42 | q | 58 | 6 |
| 11 | L | 27 | b | 43 | r | 59 | 7 |
| 12 | M | 28 | c | 44 | s | 60 | 8 |
| 13 | N | 29 | d | 45 | t | 61 | 9 |
| 14 | O | 30 | e | 46 | u | 62 | + |
| 15 | P | 31 | f | 47 | v | 63 | / |

### 工作流程

通过流程图3.2。当用户需要针对三维模型进行做自动语义标准的时候。首先用户需要先上传三维模型进该系统。接着通过调用系统中所设计的三维模型转换二维图像功能。通过HTML Canvas上处理WebGL技术，该导入的三维模型将会在各个角度被自动截图。该截图将为我们所需要转换并且进行图形分析比较相似度。

首先将三维模型所转换的二维图像通过二维图形识别处理功能，可以得到这些图像的基本标签定义以及基本的语义标注。系统会自动根据这些所得到的语义标注为原来导入的三维模型赋值。 当三维模型拥有了语义标注信息之后。可对其进行分类，聚类模型操作。从该语义标注方法的处理过程可看到比较模型相似的自动语义标注方法的输入资源基本是从二维图像识别自动语义标注方法得到的结果。 之后通过访问数据库，在海量模型数据中找出类似同样的模型数据类型。得到更为相关性的模型数据。然后进行原三维模型与其他同类模型的相似度比较。再次进行语义标注，同步相似度比较高的模型之间的语义信息。

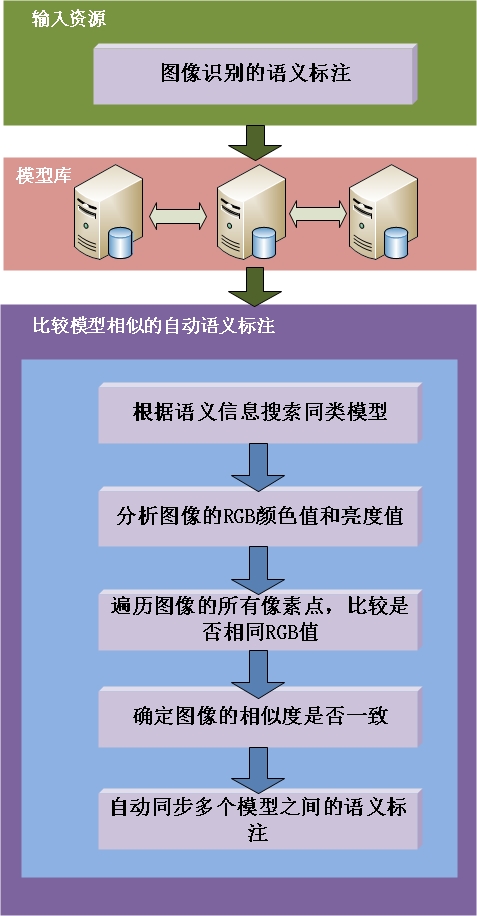


图3-5比较模型相似的自动语义标注

Fig.3-5 Comparison of the model is similar to automatic semantic annotation

## 图像的识别的语义标注方法



### 三维模型转换二维图像

### 二维图像的信息存储

系统的识别模块能够正常运作之前需要有至少一个签名作为样本。因此使用系统时第一工作需要执行是记录自己的签名样本。系统需要记录的对象有两种：完整签名和该签名中每个笔画元素。系统的记录工作也包含两种：记录至数据库及存储签名的图像文件。

本系统展示给手机用户的是完整的签名，这类型的对象可应用在样本管理方面但不用于对比功能。相反，签名的各笔画类型将不向用户显示，也不用于管理模块但是比较模块最关键的元素。因此在记录签名作为样本的过程中这两种对象都是不可缺少的。

数据库及文件管理系统将担任签名记录过程的所有工作，且两者有很密切的关系。当用户设置属于自己的签名样本时，整个签名和签名的各笔画的属性会同时被记录至数据库和以PNG格式文件存至系统的样本管理路径内。

基于数据库类型的数据在记录时针对三种对象：签名、笔画和错误类型。其中，设置属于签名和笔画这两种对象的数据表能保证两者之间有相关联系的关系也是关键的工作之一。由于基于签名的认证系统使用的密码为图像型，密码本身还未有属于自己的唯一性，因此需要手动给签名和笔画设置唯一性的属性，也就是独一的id。使用签名的id和访问到属于该签名的所有笔画，于是在笔画的关系数据表中需要所属的签名id属性。不仅这样，能够记录所画出每笔画的持续时间也有很大的意义。目前系统还未考虑各笔画的持续时间作为对比的元素，但在未来的后期工作可利用获取到的这些持续时间作为签名的对比元素这一，从而增强系统的可靠性。签名和笔画都需要获取记录时的系统时间使用户容易观察及管理，签名对象还可以添加自定义名字的属性，从而使该数据表更有人性化。关于错误类型，此对象在认证的时候会使用到。每当用户所输入的签名不符合需求，导致验证失败，系统将记录下错误所属的类型和发生的时间。错误类型数据表能使手机用户每次访问手机能查询出验证出错信息，找出存在的不异常的验证错误从而推出手机是否被他人访问过的可能性。该数据表的数据还可在汇报统计中使用到，从错误类型的数据所画出的曲线表可让用户能更加清晰的观察到在一定时间段内的验证错误情况。从以上的分析，可推出签名、笔画和错误类型数据表拥有的属性：

### 工作流程

通过流程图3.2。当用户需要针对三维模型进行做自动语义标准的时候。首先用户需要先上传三维模型进该系统。接着通过调用系统中所设计的三维模型转换二维图像功能。通过HTML Canvas上处理WebGL技术，该导入的三维模型将会在各个角度被自动截图。该截图将为我们所需要转换并且进行图形分析比较相似度。

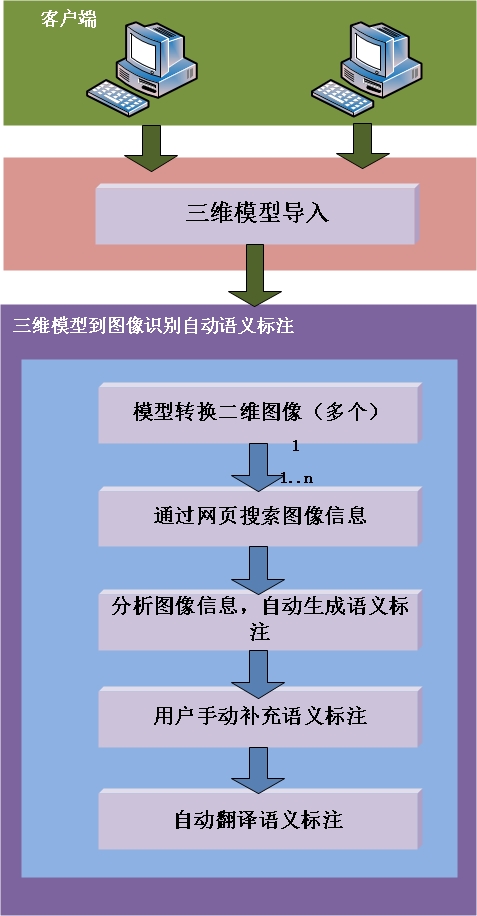


图3-6图像识别自动语义标注

Fig.3-6 Automatic semantic annotation of image recognition

## 本章小结

# **视频资源的语义标注**

本章将提出面向WebGL的自动化语义标注过程的总体框架。首先，我们将通过若干应用场景的分析来引导出标注框架的需求，并据此提出本文所研究的标注框架。随后，本章将介绍该框架内执行三维场景语义标注的一个典型流程。框架中的主要部件及它们所起的作用将在这之后得到简要的描述。



## 背景

近年随着三维文档的受欢迎程度不断提高，越来越多的应用中涉及到了针对三维场景的查询和检索。一个常见的应用场景例如一个使用工业零件三维模型库的制造业企业，它希望能根据特

## 基于Web的视频播放器

### 设计目标

在该章节中主要介绍如何实现基于浏览器窗口中内嵌入一个能够支持AVI, MPEG等视频资源格式的音视频播放器。使用户可以在上网冲浪的同时在线观看视频。当用户在点击多媒体链接时, 可以向服务器发出数据请求, 然后与视频服务器建立连接,接收服务器传送的视频数据,并且调用播放器进行解码播放。实现基于Web浏览器中视频播放器, 要考虑到视频播放架构高服务质量，低延时的要求。

### 模块划分

基于上述设计目标,可以将架构分为几个主要部分,即浏览器模块、解码播放模块、播放器内嵌浏览器模块以及流媒体服务器模块。浏览器模块主要是实现用户的浏览网页、向服务器请求数据并接收服务器的响应信息的功能。播放模块主要是实现音视频的解码播放。播放器内嵌模块主要是实现浏览器与播放器之间的连接,即实现在浏览器接收到服务器的响应信息后可以调用播放模块进行解码播放。流媒体服务器模块主要是存放压缩后的音视频信息,并且通过网络以流的方式向客户端浏览器传送音频，视频，多媒体文件。

将总体架构进行模块划分后,课题的设计结构更加明朗。课题设计了解码播放模块和播放器内嵌浏览器模块, 浏览器模块采用了嵌入式系统广为采用的火狐浏览器。解码播放模块采用Gstreamer架构进行设计。该开源多媒体开发框架将由下面的内容详细介绍并讨论的特点以及作用。

### Gstreamer多媒体框架

* + - 1. 框架介绍

GStreamer 是 GNOME 桌面环境下用来构建流媒体应用的开源多媒体框架（framework），其目标是要简化音/视频应用程序的开发，目前已经能够被用来处理像 MP3、Ogg、MPEG1、MPEG2、AVI、Quicktime 等多种格式的多媒体数据。

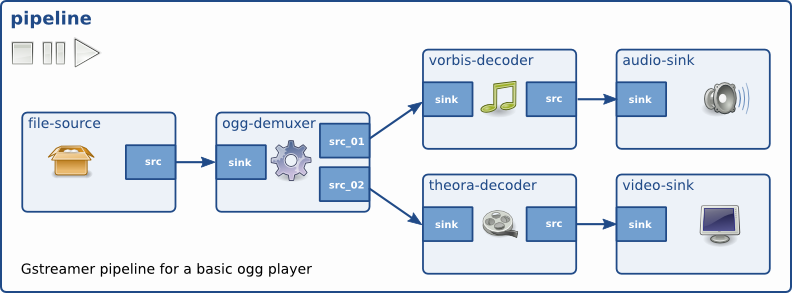


图4-7 Gstreamer 流水线实现OGG播放器

Fig.4-7 Gstreamer pipeline for a basic OGG player

GStreamer是一个开源的多媒体框架库。利用它，可以构建一系列的媒体处理模块，包括从简单的OGG播放功能到复杂的音频（混音）和视频（非线性编辑）的处理。基于Gstreamer流水线实现OGG播放器功能可见图。应用程序可以透明的利用解码和过滤技术。开发者可以使用简洁通用的接口来编写一个简单的插件来添加新的解码器或滤镜。

* + - 1. Gstreamer框架的特点

GStreamer是一个创建流媒体应用程序的框架。其基本设计思想来自于俄勒冈(Oregon)研究生学院有关视频管道的创意, 同时也借鉴了DirectShow的设计思想。

GStreamer的程序开发框架使得编写任意类型的流媒体应用程序成为了可能。在编写处理音频、视频或者两者皆有的应用程序时, GStreamer可以让你的工作变得简单。GStreamer并不受限于音频和视频处理, 它能够处理任意类型的数据流。管道设计的方法对于实际应用的滤波器几乎没有负荷, 它甚至可以用来设计出对延时有很高要求的高端音频应用程序。具体Gstreamer多媒体核心框架是如何实现的由以下的架构图所示。

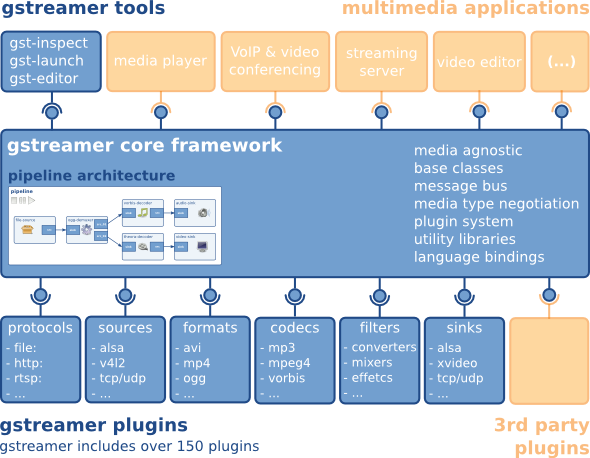


图4-8 Gstreamer 核心框架架构图

Fig.4-8 Gstreamer core framework architecture diagram

GStreamer最显著的用途是在构建一个播放器上。GStreamer已经支持很多格式的文件了, 包括: MP3、Ogg/Vorbis、MPEG-1/2、AVI、Quicktime、 mod等等。从这个角度看,GStreamer更象是一个播放器。但是它主要的优点却是在于: 它的可插入组件能够很方便的接入到任意的管道当中。这个优点使得利用GStreamer编写一个万能的可编辑音视频应用程序成为可能。

GStreamer框架是基于插件的, 有些插件中提供了各种各样的多媒体数字信号编解码器,也提供了其他的功能。所有的插件都能够被链接到任意的已经定义了的数据流管道中。GStreamer的管道能够被GUI编辑器编辑, 能够以XML文件来保存。这样的设计使得管道程序库的消耗变得非常少。

GStreamer核心库函数是一个处理插件、数据流和媒体操作的框架。 GStreamer核心库还提供了一个API, 这个API是开放给程序员使用的---当程序员需要使用其他的插件来编写他所需要的应用程序的时候可以使用它。

### 工作模式

随着网络功能的强大, 视频播放也变化出多种工作方式。用户在点击网页上的多媒体链接, 服务器与客户端浏览器便会建立一个请求响应连接, 浏览器在检查多媒体链接说明信息后便会调用播放器进行解码播放。这样, 服务器、浏览器、播放器之间便形成了紧密的联系。考虑到延时和复杂度的问题, 基于Web HTML5的视频播放器与服务器和浏览器之间的工作模式是设计所要考虑的一个主要问题。因此, 本文设计了一种将视频播放器内嵌到浏览器中与Web服务器相连的工作模式, 下面的图说明了视频播放的工作流程和架构图。

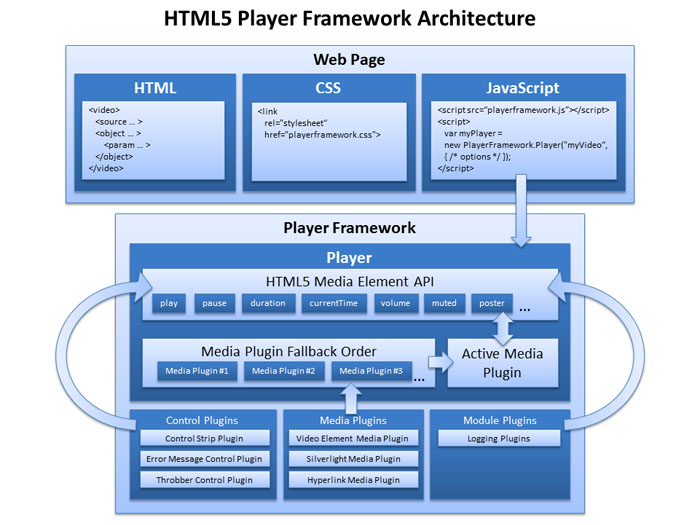


图4-9 基于Web的视频播放器架构图

Fig.4-9 Web-based video player architecture diagram

基于Web HTML5的视频播放器采用如上所示的工作模型, 这种工作模式可以采用如下的方法实现。首先, 当用户点击网页上的视频音频文件的链接请求传送文件, 建立与服务器的连接, 链接会指向一个包含文件URL的一个播放说明文件, 并且将这个URL地址封装在服务器响应的HTML消息中。浏览器接收到这个播放说明文件并对其进行解析, 检查通过HTTP协议的响应消息中的音视频文件的消息内容和媒体数据格式, 调用内嵌在浏览器中的视频播放器。最后, 播放器对音视频文件进行解码播放。

## 视频的自动截取画面

## 视频的语义标注

## 本章小结

# 系统实现与验证

本章将首先给出原型系统的总体架构，随后对其中的部分模块的实现细节进行讨论。本章的后半部分将通过一个具体的应用实例来介绍如何使用我们的系统，并由应用的结果来验证本文所提出方法的实用性和有效性。

## 自动语义标注框架和总体架构

根据第二章所述的三维模型语义标注总体框架和设计研究思想。由以下图5-1三维模型自动语义标注整体以及整个三维模型建模平台总框架所示。

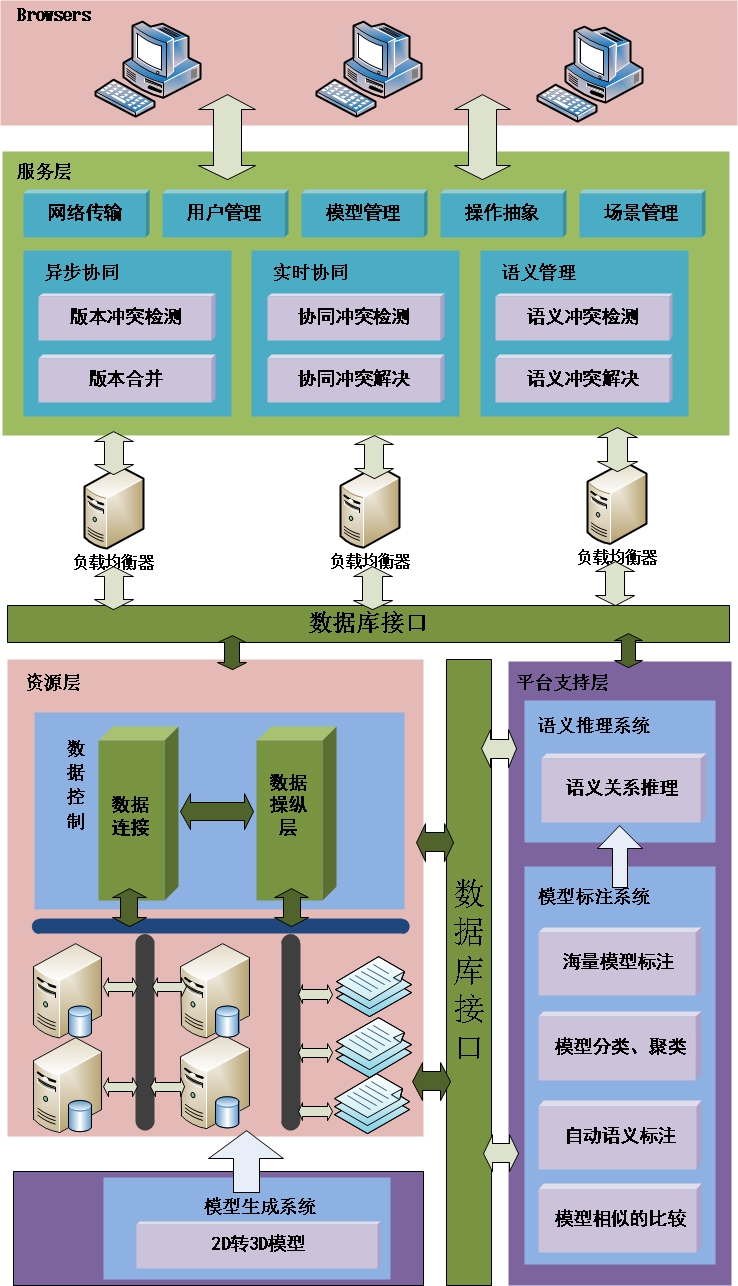


图5-1三维模型建模平台的架构

Fig. 5-1 Three-dimensional modeling platform architecture

## 三维模型自动语义标注的设计过程

本系统使用网络应用程序设计语言HTML、CSS、JavaScript与jQuery来实现系统界面与内部算法。该三维模型的自动语义标注系统后台使用PHP 开发语言Yii MVC开发环境进行开发系统。大部分系统的业务逻辑都在后台的Model层实现。系统实现所使用到的数据库选用的是MongoDB。由于该系统的前后台数据访问都是为JSON数据格式。而且MongoDB专门设计支持该数据结果，以便存储语义信息更加方便。

### 三维模型的导入和展示

这部分主要将介绍该系统的三维模型的展示部分以及介绍基于网页前端中如何实现通过渲染器将三维模型展现和处理。系统的三维模型展示页面有由以下图5-2所示。

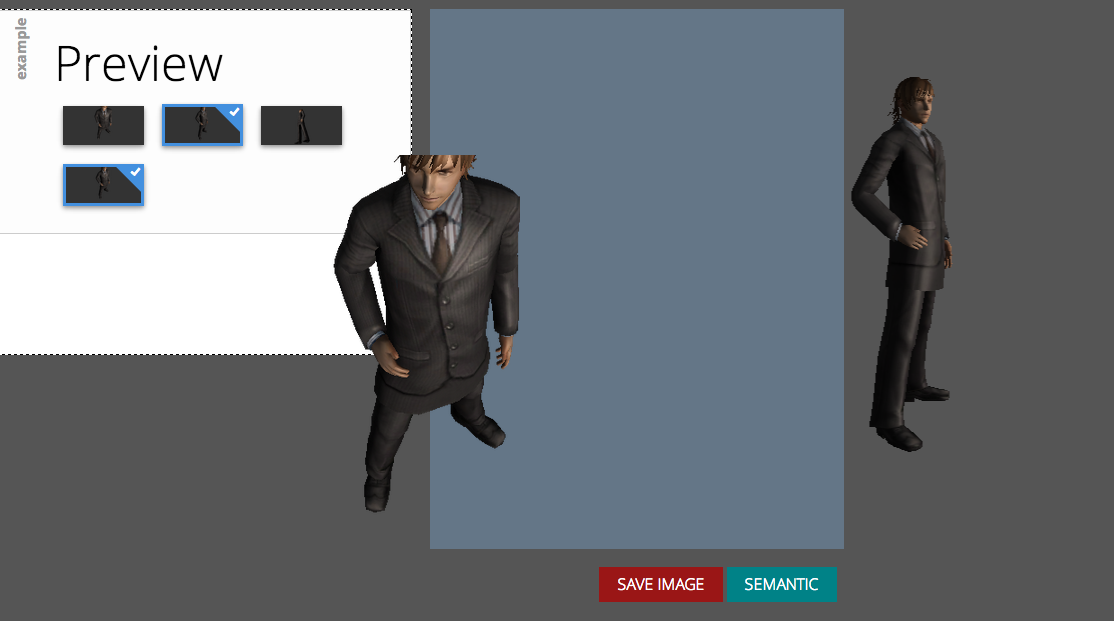


图5-2三维模型的渲染展示效果图

Fig. 5-2 the effect of rendering three-dimensional model diagram

在本文中HTML5以及CSS 3.0主要作为前端设计的语言，作为MVC中展示层面的实现方法。在三维模型的展示部分以及三维模型自动语义标注，前端的二维图像分析和相似度比较部分都采用该语言作为设计方法。HTML是超文本标记语言，作为本系统的界面设计语言。该语言无需使用编译器，在编写HTML文件后可直接使用浏览器预览并测试与使用。使用该语言使系统设计工作容易管理及处理错误。HTML界面结构也集成了系统的所有逻辑库、设计风格而展示出系统功能与结构。该部分可视为用户与系统之间的交互接口。 最新的HTML5可支持canvas图像处理，该性能是本系统的关键模块之一。

CSS是串样式列表语言，该语言负责向HTML文件提供布局、彩色及字体设置。原始，HTML即可通过<style> </style>标签来存放某界面元素的设计风格。但当文件数过多，需要统一修改或更新时需要对所有HTML档案进行修改，该操作将花费很多时间及统一性很低，容易导致错误或不一致性。因此使用CSS语言可解决该问题。将整系统的设计风格存放在一个CSS列表文件中可使对系统修改或更新变得方便、节省时间且有高统一性与效率性。

表5-1三维模型WebGL的渲染展示代码

**function** init() {

// 基于WebGL的三维场景摄像头设置

camera = **new** THREE.PerspectiveCamera(45, window.innerWidth / window.innerHeight, 1, 2000);

    camera.position.z = 100;

    // 基于WebGL的三维场景构建

    scene = **new** THREE.Scene();

**var** ambient = **new** THREE.AmbientLight(0x444444);

    scene.add(ambient);

// 三维场景渲染的直射灯光效果设置

**var** directionalLight = **new** THREE.DirectionalLight(0xffeedd);

    directionalLight.position.set(0, 0, 1).normalize();

    scene.add(directionalLight);

    // 3D Model Importing

**var** loader = **new** THREE.OBJMTLLoader();

// 从数据库中加载需要展示的三维模型文件

 loader.load('obj/male02/male02.obj', 'obj/male02/male02\_dds.mtl', **function**(object) {

        object.position.y = -80;

        scene.add(object);

    });

    // 基于Three.js的WebGL三维模型的渲染器设置

    renderer = **new** THREE.WebGLRenderer({

        alpha: **true**,

        preserveDrawingBuffer: **true** // required to support .toDataURL()

    });

    renderer.setSize(window.innerWidth / 1.5, window.innerHeight / 1.5);

}

本文的系统设计中，网页前端除了使用HTML5和CSS 3.0以外。大部分的前端逻辑代码实现都采用了JavaScript脚本语言以及JavaScript代码框架JQuery。JavaScript是一种动态脚本语言，经常用于网络应用程序设计。该语言作为本系统所有内置逻辑算法的实现工具。JavaScript支持基于面向对象程序设计及动态类型转换。有许多开源软件库能提供给JavaScript使用，使该语言变的更灵活及高效率。JavaScript实现的功能将以HTML界面作为借口来展示。本系统的大部分前端处理的逻辑代码，前端美工以及页面效果，三维模型渲染器，三维模型展示效果，图像识别以及二维图片相似度比较算法部分都是由JavaScript实现。

该框架是基于JavaScript的跨平台软件库。目前jQuery为JavaScript最广用的开源库。jQuery可提高JavaScript对HTML元素的操作及事件处理。该库使编写JavaScript代码工作变得更加简洁、容易操作和方便修改。该框架的使用目的为了减少前端开发的工作量以及提高前端逻辑代码处理的性能。让代码更加规范而提高可维护性，代码继承和扩展性。

通过以上所使用的前端设计方法，可实现在网页上基于WebGL技术通过HTML5 Canvas将三维模型渲染效果。不但如此，JavaScript逻辑代码可监控用户移动鼠标行为获取到鼠标位置。实现三维模型根据鼠标的位置来旋转场景的摄像头角度效果。三维模型展示效果如图5-3所示。

|  |  |  |
| --- | --- | --- |
| E:\GitHub\semantic\doc\graph\3d\5385d016a1313.png | E:\GitHub\semantic\doc\graph\3d\538215adbffef.png | E:\GitHub\semantic\doc\graph\3d\test.png |

图5-3三维模型各个角度的渲染效果

Fig. 5-3 rendering three-dimensional model from different angles

本文中所使用的三维模型大部分都是基于WebGL的技术。而这样的模型文件需要有渲染器才可以展示在网页上。由于考虑网页上的3D技术还有比较多的限制和瓶颈。所以需要使用较好的渲染算法为了提高浏览器上三维模型的加载速度和处理性能。本文中采用的是Three.js是目前最为广泛的浏览器上3D渲染器。该三维渲染器引擎是JavaScript编写的WebGL第三方库。提供了非常多的3D显示功能。Three.js 是一款运行在浏览器中的 3D 引擎，你可以用它创建各种三维场景，包括了摄影机、光影、材质等各种对象。你可以在它的主页上看到许多精彩的演示。不过，这款引擎目前还处于非稳定版本，不成熟的开发阶段。但在性能上已经满足简单的三维模型渲染效果。

Three.js的三维模型的渲染器允许创建使用JavaScript语言作为一个网站的一部分，GPU加速3D动画，而无需依赖于专有的浏览器插件。这也意味着该插件完全利用了HTML5所提供的WebGL渲染技术。

表5-2三维模型的渲染展示效果输出代码

   //PREVIEW

    $("#preview").css({

        height: (window.innerHeight / 2) + "px"

    })

  container.appendChild(renderer.domElement);

  // 在浏览器的三维模型展示页面中监听鼠标移动位置

    document.addEventListener('mousemove', onDocumentMouseMove, **false**);

    // 在浏览器的三维模型展示页面中监听窗口的尺寸大小变化

    window.addEventListener('resize', onWindowResize, **false**);

  }

// 三维模型的渲染器函数

**function** render() {

    camera.position.x += (mouseX - camera.position.x) \* .05;

    camera.position.y += (-mouseY - camera.position.y) \* .05;

    camera.lookAt(scene.position);

    renderer.render(scene, camera);

}

// 监听浏览器窗口的鼠标位置以及状态

**function** onDocumentMouseMove(event) {

    mouseX = (event.clientX - windowHalfX) / 2;

    mouseY = (event.clientY - windowHalfY) / 2;

### 三维模型自动转二维图像方法实现

本文中三维模型所转换的二维图像也是一个非常重要，置于系统中较为核心部分的位子。该方法也是本文所提出的一个创新点。通过三维模型转换成二维图像后，该系统可以根据所转换的二维图像进行自动语义标注或者将该图像进行分析图像特征(颜色，亮度，等等)。从而可得到数据库中类似的三维模型资源。通过该方法，一个导入的三维模型在不同的角度和尺寸可以转换到多个二维图像。该功能的界面由以下图5-4所示。

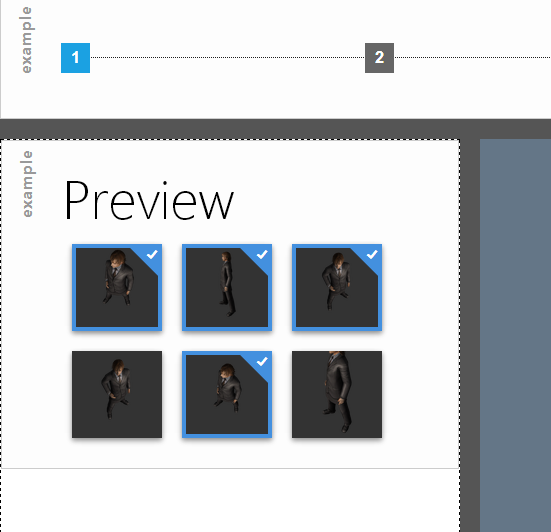


图5-4三维模型转换二维图像基于Base64码

Fig.5-4 Three-dimensional model conversion based on Base64 code

本文中的三维模型转换的二维图像功能有表5-3中的代码所实现。通过实现代码可看出三维模型的转换方法主要是利用于HTML5 Canvas技术。由以上5.3.1中所提到的三维模型在浏览器中通过Three.js渲染器引擎展示出来。三维模型通过渲染器后可在Canvas标签中展示成二维平面效果。也意味着用户最后所看到的每一帧 (frame rate) fps在浏览器并非是真正三维模型，而是已经被渲染器处理后所产生的二维图片效果。

利用以上所述的结论，本文将通过JavaScript代码实现一个三维模型转换二维图像方法。一下代码中的toDataURL ()函数就是该方法的核心部分。在该函数中需要提供的参数应该是渲染器引擎Three.js所定义的renderer变量，其次是需要最后转换的类型。在默认情况程序将会转换成基于Base64编码的png/Image格式。由函数中的代码renderer.domElement.toDataURL(mimetype); 所实现。具体的实现方法可见以下5-3表的代码所述。

表5-3三维模型WebGL转换Base64码的二维图像

/\*\*

 \* Take a screenshot of a renderer

 \* - require WebGLRenderer to have "preserveDrawingBuffer: true" to be set

 \*

 \* @param {Object} renderer to use

 \* @param {String} mimetype of the output image. Default to "image/png"

 \* @param {String} dataUrl of the image

 \*/

**var** toDataURL = **function**(renderer, mimetype) {

    mimetype = mimetype || "image/png";

**var** dataUrl = renderer.domElement.toDataURL(mimetype);

**return** dataUrl;

}

  // export it

//主要程序。将自动三维模型转二维图像封装

THREEx.Screenshot = {

    toDataURL: toDataURL,

    bindKey: bindKey

  };

通过上述表5-3所示的代码，可以实现将一个三维模型转换为多个二维图像方法。通过该方法，三维模型可转换到各种常见的图像文件格式。在浏览器上使用Base64编码预存储。为了考虑之后方便实现在三维场景下，模型是可以通过各个角度进行自动的转换为二维图像。本文中所实现一个事件绑定的函数。该函数作用就是将表5-3中的代码可以绑定与用户可配置的热键。通过调用在绑定事件，三维模型在三维场景下摄像头的各个角度可以任意将其截取对应的二维图像。

当用户进行三维模型转换多个二维图像的时候，可配置参数是否需要将输出的二维图像进行尺寸调节。若用户不需针对二维图像的尺寸做调整时，该程序将会按照原来三维模型所被渲染在HTML Canvas中大小作为默认的二维图像输出的尺寸大小。将输出的Base64图像文件编码作为调回函数*callback(dataUrl)* 的参数。其中*dataURL*就是二维图像的内容。如果用户需要对二维图像尺寸做定义调整时，该转换器中的代码实现也提供一个智能的二维图像尺寸修剪功能。

通过函数*aspectResize(dataUrl, width, height, callback)* 所实现。该功能保证在用户所自定义的输出二维图像尺寸大小，该二维图像依然能够按照原来的三维模型尺寸的比例而输出结果。使得所转换的二维图像跟原来渲染的三维模型尺寸比例相等，保持所输出二维图像没有变异性。具体该功能的具体实现方法可见以下表5-4中的实现代码所示。

表5-4三维模型WebGL转换二维图像的事件绑定

  /\*\*

 \* Bind a key to renderer screenshot

将三维模型转换二维图像的绑定事件函数

 \*/

**var** bindKey = **function**(renderer, opts) {

    // handle parameters

    opts = opts || {};

**var** charCode = opts.charCode || 'p'.charCodeAt(0);

**var** width = opts.width;

**var** height = opts.height;

**var** callback = opts.callback || **function**(url) {

        window.open(url, "name-" + Math.random());

    };

    // callback to handle keypress

**var** onKeyPress = **function**(event) {

        // return now if the KeyPress isnt for the proper charCode

**if** (event.which !== charCode) **return**;

        // 通过定义的渲染器 将三维模型转换截图

**var** dataUrl = **this**.toDataURL(renderer);

**if** (width === undefined && height === undefined) {

            callback(dataUrl)

        } **else** {

            // resize it and notify the callback

            // \* resize == async so if callback is a window open, it triggers the pop blocker

//二维图像智能调整大小函数

            \_aspectResize(dataUrl, width, height, callback);

        }

    }.bind(**this**);

//将三维模型截图事件绑定于设置的热键

// NOTE: for firefox it seems mandatory to listen to document directly

    document.addEventListener('keypress', onKeyPress, **false**);

**return** {

// 解除绑定热键

        unbind: **function**() {

            document.removeEventListener('keypress', onKeyPress, **false**);

    }

  };

}

通过以上表5-3的代码所描述的三维模型转换的二维图像功能。在浏览器上最后可得到基于base64的文件编码内容。但问题这些图片还只是处于前端浏览器，而没有被永久的存储下来。为了解决这个问题，本文中设计的系统有一个二维图像的存储功能模块。主要实现方法是由前端浏览器通过Ajax请求，将存于前端缓存的二维图像由Base64编码格式发送的后台。基于Base64编码的二维图像文件格式以正常的字符串传送。再由后台处理相应的请求然后存到本地服务器上。对应的图片信息也会存储于MongoDB数据库里面，为了之后方便查找和管理。具体二维图像的发送请求以及相关参数由以下的列表所示。

前端浏览器POST请求发送二维图像到服务器：

发送URL 地址：<http://semantic/index.php?r=site/saveimage>

|  |  |  |  |
| --- | --- | --- | --- |
| **标签名** | **数据类型** | **必选** | **说明** |
| Action | 正整数 | 是 | 前端所调用后台的触发事件 |
| Image | 字符串 | 是 | 二维图像文件内容。主要为基于base64的文件编码内容 |
| Path | 字符串 | 是 | 二维图像所保存的本地路径。数据为字符串 |
| Type | 字符串 | 否 | 图像文件存储到本地服务器的格式。如果参数为空的话，默认存储为PNG文件格式 |

后台服务器返回结果实例（JSON数据格式）

{

"Result": "images\/5491a3e8a715b.png",

“Status”: 200

}

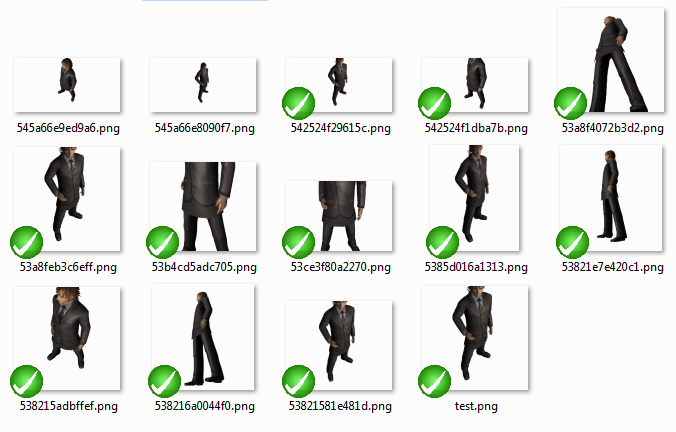


图5-5三维模型转换二维图像的最终文件

Fig.5-5 Three-dimensional model conversion to finally image files

通过前端浏览器的Ajax请求将二维图像的文件内容发送到后台服务器。该后台将会有一个存储器程序将接受内容原本是Base64编码的文件格式转换为正常所见到的二维图像文件格式（如png，jpg文件等等。。。）。然后将该文件写入近本地的服务器磁盘里面。该程序的操作流程由以下表5-5中的代码所描述。

表5-5三维模型转二维图像的本地存储函数实现代码

//图像的本地存储函数实现

**public** **function** saveImage($dir) {

            //定义文件上传的目录路径

            define('UPLOAD\_DIR', $dir);

            $img = $this-> imgURL;

           //将二维图像从Base64编码格式转换为常见的图像文件格式（默认PNG文件）

            $img = str\_replace('data:image/png;base64,', '', $img);

            $img = str\_replace(' ', '+', $img);

            $data = base64\_decode($img);

             //UNIQUE ID 产生唯一的文件ID号码

             $id = uniqid();

            $file = UPLOAD\_DIR . $id . '.png';

            //如果没有对应的文件夹则新建一个

**if**(file\_exists (UPLOAD\_DIR) == false)

                mkdir(UPLOAD\_DIR);

            //写入图像的文件到本地服务器

            $success = file\_put\_contents($file, $data);

            //返回文件写入状态是否成功

**return** $success ? $file:"";

        }

### 自动识别图像的语义标注方法实现

这部分主要将介绍本文中系统实现的三维模型自动语义标注的图像识别方法。该方法的设计与实现主要分成两部分。前端可视化操作层以及后端的服务器的业务逻辑处理。在该方法中主要的输入资源来自于5.2部分所陈述的三维模型所转换的二维图像资源。通过后台的对二维图像的识别和处理图像信息后，结果会返回给前端通过Ajax异步请求。数据返回均为JSON数据格式。在一般如果后台服务器能够识别出三维模型所转换的二维图像信息时，会返回相应的语义标注结果以及一些标签信息。

系统的后台服务器经过处理图像识别所得到的语义标注信息后。该语义信息会被发送往回给前台浏览器。浏览器通过JavaScript语言实现的数据处理模块后。所有获取到三维模型语义标注信息会被渲染成可视化的HTML代码。信息展现在浏览器界面上如图5-6所示。



自动提示标注

用户自定义标签

语义标注 tags

图5-6基于互联网搜索的自动语义标注图

Fig.5-6 Internet-based automated semantic annotation search graph

在前端浏览器的操作界面，系统会将需要识别的二维图像通过AJAX异步请求发送到后台服务器。具体的请求发送参数Schema可以参考一下的列表

前端浏览器POST请求发送二维图像到服务器进行图像识别处理：

发送URL 地址：<http://semantic/index.php?r=semantic/getSemanticImage>

|  |  |  |  |
| --- | --- | --- | --- |
| **标签名** | **数据类型** | **必选** | **说明** |
| Action | 正整数 | 是 | 前端所调用后台的触发事件。默认触发Controller的getSemanticImage事件 |
| Image | Base64编码 | 是 | 二维图像文件内容。图像的文件格式基于base64的文件编码内容 |
| Source | 字符串 | 是 | 二维图像的来源。可以是从互联网的URL图像地址。或者从本地服务器的文件路径。默认为from local storage |
| Path | 字符串 | 否 | 二维图像存储在本地服务器的文件路径 |

后台服务器返回对应三维模型的语义标注结果实例（JSON数据格式）。语义信息结果在*Keyword*标签里面所示

{

"Keyword": "("枭龙\*战机\*隐身\*改进型\*大阅兵\*空军论坛\*舰载机\*神秘\*"),",，

“imgQuerySign” :"1358363483,2885141759",

“Status”: 200

}

在本文的系统设计中，该功能的所有主要业务逻辑处理都是处于后台服务器处理。后台使用PHP开发语言结合Yii 框架的MVC开发模式来开发。Yii Framework是一个基于组件、用于开发大型 Web 应用的高性能 PHP 框架。该框架提供了今日Web 2.0应用开发所需要的几乎一切功能。它是最有效率的PHP框架之一。

系统设计实现所用的Yii框架是一个基于组件的高性能PHP框架，用于开发大型Web应用。Yii采用严格的OOP编写，并有着完善的库引用以及全面的教程。从 MVC，DAO，widgets，caching，等级式RBAC，Web服务，到主题化。该框架提供了目前Web 2.0应用开发所需要的几乎一切功能。

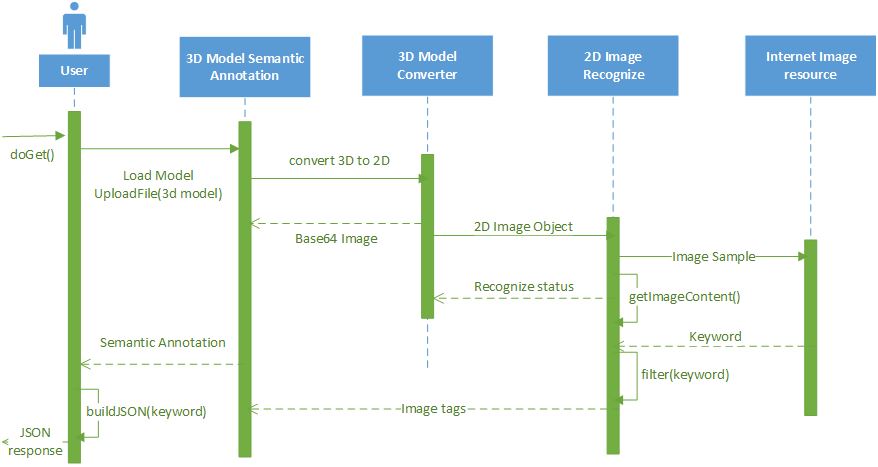


图5-7基于互联网搜索的自动语义标注时序图

Fig.5-7 Internet-based automated semantic annotation sequence diagram

后台服务器识别和处理二维图像信息流程按照以上时序图5-7所示。首先用户通过UploadFile(3d Model)方法将需要进行语义标注的三维模型加载到系统中。该系统会通过三维模型转换二维图像功能模块，将该模型进行转换通过converter(3D model)方法。三维模型所转换的二维图像结果会以Base64编码文件格式返回到浏览器界面上。浏览器通过JavaScript逻辑代码实现，将该二维图像的文件打包转换成图形对象。然后通过AJAX请求将该对象发送到后台服务器，让后台服务器将该二维图像进行识别信息。

后台服务器处理二维图像识别的具体方法按照这样的步骤。首先后台接受到前端浏览器发送过来的二维图像后，需要对其进行获取二进制图像的源码。代码中通过*file\_get\_contents($image)* 方法可实现该功能。当二维图像的二进制源码获取到后。将其封装在一个表单的形式，作为其中的一个参数。然后将整个模拟表单通过Curl()方法提交到互联网上的一些图片检索服务器（如百度，谷歌，等等。。。）进行搜索该二维图像的类似图片以及相关的网页内容。该请求会在服务器的底层处理并返回结果到服务器应用层。可通过*curl\_exec($ch)* 方法获取到从互联网搜索的图片信息。

本文的系统设计实现后台服务器中，有部分代码会利用到Curl()方法来获取来自互联网的数据。本身cURL是一个利用URL语法在命令行下工作的文件传输工具。它支持文件上传和下载，所以是综合传输工具，但按传统可将cURL扩展模块视为一个下载工具。cURL还包含了用于程序开发的libcurl。

以上的后台服务器处理二维图像识别方法的代码实现由以下表5-6中的代码实现所述。

表5-6基于互联网搜索的自动语义标注

/\*SEMANTIC FROM FILE\*/

**public** **function** getSemanticFromFile()

    {

        //图片路径

        $image = $this->getImgURL();

        //图片的二进制源码

        $img = file\_get\_contents($image);

 //发送至识别图像服务器的主要参数

        $param = **array**(

            "client" => "pcbrowser",

            "tn" => "baiduimage",

            "id" => "WU\_FILE\_0",

            "name" => "semantic.jpg",

            "type" => "image%2Fpng",

            "size" => 0

        );

        $url = $this->getHost() . "?" . http\_build\_query($param);

               // create curl resource

        $ch = curl\_init();

          // set url

        curl\_setopt($ch, CURLOPT\_URL, $url);

        curl\_setopt($ch, CURLOPT\_POSTFIELDS, $img);

        //return the transfer as a string

        curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

         // $output contains the output string

        $output = curl\_exec($ch);

          // close curl resource to free up system resources

        curl\_close($ch);

**return** trim($output);

    }



自动推荐同类图

语义标注的定义

图5-7基于互联网搜索的自动推荐图

Fig.5-7 Internet-based searching automatically recommends image

表5-7基于互联网搜索结果自动推荐相似二维图片代码实现

//INIT SEMANTIC

**public** **function** getSimilarImage($querysign)

    {

        //对应请求的参数

        /\* $querysign

        根据 Query 字符串来搜索相应的同类相似度图片

         \* \*/

        $param = **array**(

            "ct" => 3,

            "tn" => "insimijson",

            "pn" => 0,

            "rn" => 20,

            "shituRetNum" => 100,

            "similarRetNum" => 600,

            "querysign" => $querysign

        );

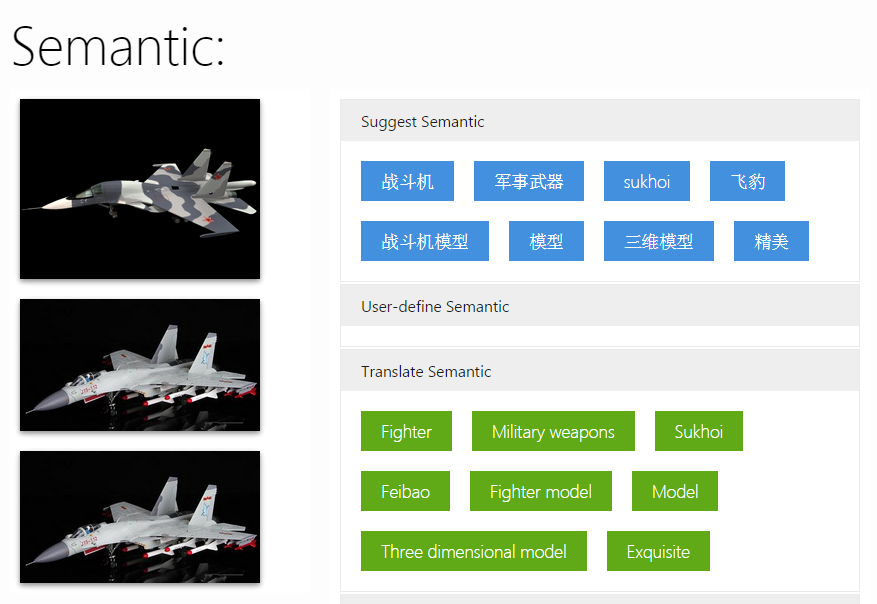
        $url = $this->host . "?" . http\_build\_query($param);

        $result = file\_get\_contents($url);

**return** $result;

        //echo implode("&", $param);

    }



语义标注 tags

语义标注的自动翻译

图5-8三维模型语义标注的自动翻译模块

Fig.5-8 Three-dimensional model of semantic annotation of automatic translation module

表5-8三维模型语义标注的自动翻译模块的代码实现

//TRANSLATE ANNOTATION

**public** **function** translate($tags)

   {

       $result = **array**();

       //FILTER KEYWORD

       $filter = $this->filter($tags);

       $tags = array\_merge($tags, $filter);

       /\*FILTER KEYWORD, IN ARRAY FOR CHECK DUPLICATE \*/

       $tags = explode(",", $tags);

       $tags = implode("%0A", $tags);

       /\* BAIDU TRANSLATE API\*/

       $apikey = "2ALMz6WqUEcsBg4BS91Eppq3";

       /\* BAIDU TRANSLATE URL\*/

       $url = "http://openapi.baidu.com/public/2.0/bmt/translate?client\_id=" . $apikey . "&from=auto&to=auto&q=" . $tags;

       //GET TRANSLATE

       $data = file\_get\_contents($url);

**return** $data;

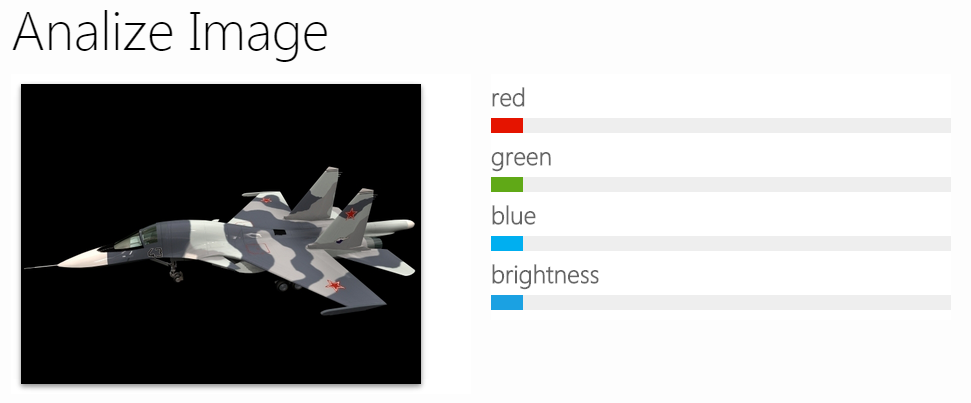
   }

## 基于前段的图像基本分析及相似度比较

### 前段的图像自动分析

图像识别是本认证系统的核心模块，工作是对从本次输入的三维模型所转换的二维图像导出的图案对象与数据库中对应样本的三维模型文件进行对比。本项目使用由James Cryer开发的Resemble作为基于HTML前段三维模型所转换的二维图像相似度比较的核心算法。以图片的像素结构作为对比目标，Resemble可识别出两幅不同格式的图片的区别点并且同时计算出两幅图的差别率。

由于本系统的对比对象是三维模型所转换的二维图像。在本文中我们进行二维图片的相似度比较时。我们主要考虑的是图片中所描述的模型转换的形状是否达到标准的相似度（本文中所定义的相似度标准值是80%）。因此在使用resemble相似度比较算法工具时无需考虑背景颜色，修边等问题。接下的实现代码解释只考虑与调用到的功能的相关的部分。

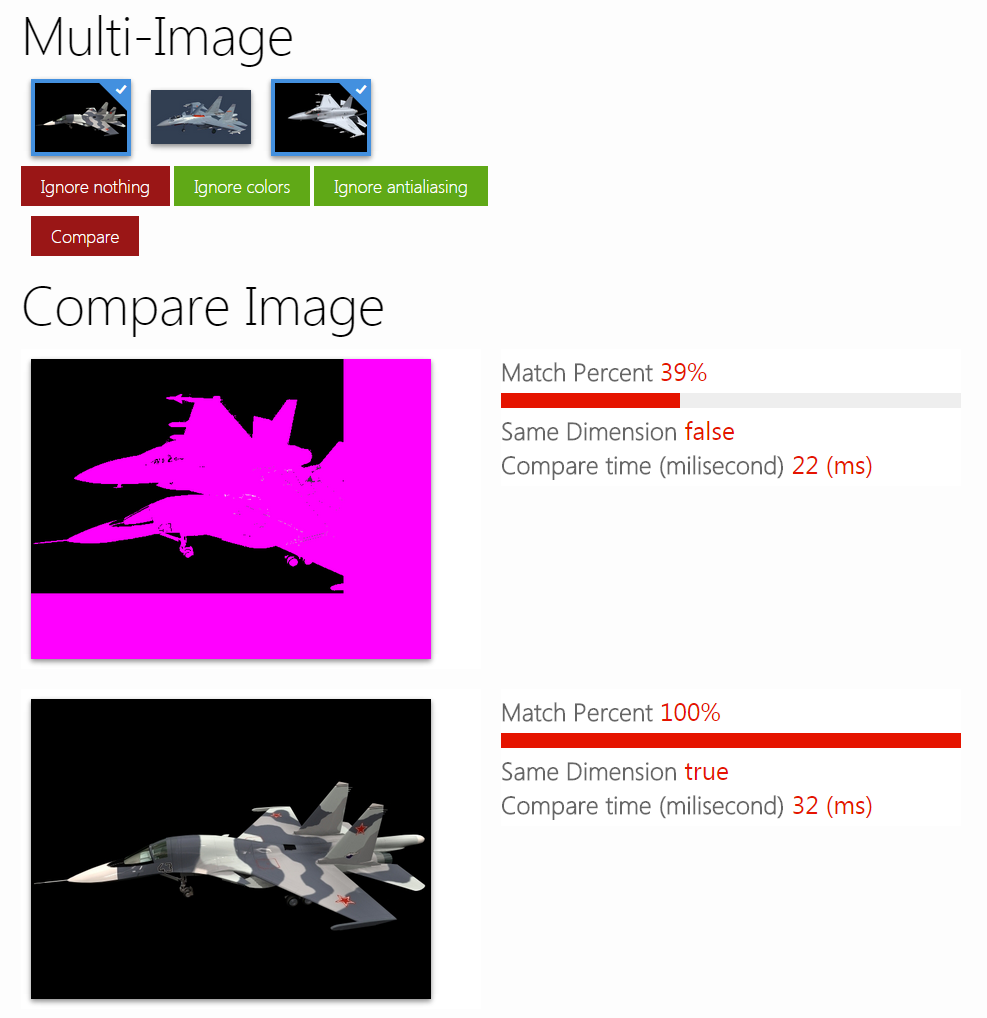


二维图像分析结果

图5-9基于前端的二维图像分析算法图

Fig.5-9 Based on the front of the two-dimensional image analysis algorithms graph

### 二维图像的相似度比较



图像比较的生成结果

数据库的同类图片

图像比较的操作按钮

图5-10三维模型语义标注的相似度比较方法

Fig.5-10 Three-dimensional model of semantic annotation similarity comparison method

首先本系统使用resemble时的调用语句resemble Control的格式如下：

表5-9基于互联网搜索结果自动推荐相似二维图片代码实现

**var** diff = resemble(file).compareTo(file2).ignoreColors().onComplete(**function**(data){

    console.log(data);

    /\* 三维模型相似度比较的配置参数

    {

      misMatchPercentage : 100, // %

      isSameDimensions: true, // or false

      dimensionDifference: { width: 0, height: -1 }, // defined if dimensions are not the same

      getImageDataUrl: function(){}

    }

});

该语句表示将第一个图片文件读入resemble中，通过compareTo函数来与第二个图片文件进行对比。对比过程始终由onComplete函数执行。整体的调用顺序是compareTo() 函数、onComplete函数、compare函数、analyseImages() 函数。接下来将解释这些函数的功能及调用。

compareTo主要是转发被使用来对比的第二图片文件对象。该函数返回的数值是以第二个对比图片对象为参数的getCompareApi对象。getCompareApi中定义了onComplete成员函数。因此原始的主调用语句中compareTo(file2).onComplete (onComplete)意味着执行包含file2对象的onComplete函数。compareTo()函数的定义如下代码所示。

表5-10二维图像相似度比较的调用函数

return {

    onComplete: function(callback) {

        updateCallbackArray.push(callback);

        loadImageData(fileData, function(imageData, width, height) {

            parseImage(imageData.data, width, height);

        });

    },

    compareTo: function(secondFileData) {

        return getCompareApi(secondFileData);

    }

};

onComplete是getCompareApi的成员函数。onComplete的主要的工作是可执行用户自定义的callback函数来处理对比过程的后续处理工作，同时通过wrapper函数调用compare函数来把两个图片对象进行对比。onComplete的函数定义如下代码所示。

表5-2二维图像相似度比较的调回函数处理

onComplete: **function**(callback) {

    updateCallbackArray.push(callback);

**var** wrapper = **function**() {

        compare(fileData, secondFileData);

    };

    wrapper();

**return** getCompareApi(wrapper);

}

compare也可视为核心的对比过程的前半部分函数。该函数主要通过onceWeHaveBoth() 函数将需要对比的两个对象的大小，包含宽度和高度，进行比较。onceWeHaveBoth() 函数先判别两者的大小是否相等。若两者的大小不匹配，则取大的数值作为下一步正常化图片的常数。所谓的正常化是向加载两幅图的canvas载体获取同样的宽度及高度大小，从而可执行对比工作。正常化后的两个图片对象会通过analyseImages() 函数来进行笔画对比工作。compare的函数定义如下代码所示。

表5-11二维图像相似度比较的核心函数调用

**function** compare(one, two) {

**function** onceWeHaveBoth() {

**var** width;

**var** height;

**if** (images.length === 2) {

            width = images[0].width > images[1].width ? images[0].width : images[1].width;

            height = images[0].height > images[1].height ? images[0].height : images[1].height;

**if** ((images[0].width === images[1].width) && (images[0].height === images[1].height)) {

                data.isSameDimensions = **true**;

            } **else** {

                data.isSameDimensions = **false**;

            }

            data.dimensionDifference = {

                width: images[0].width - images[1].width,

                height: images[0].height - images[1].height

            };

            analyseImages(normalise(images[0], width, height), normalise(images[1], width, height), width, height);

            triggerDataUpdate();

        }

    }

    images = [];

    loadImageData(one, onceWeHaveBoth);

    loadImageData(two, onceWeHaveBoth);

}

analyseImages() 函数是执行笔画对比的核心函数。该函数读入两个图片对象的数据，之后以像素的结构为单位，对两者进行扫描。每次扫描跳向下一个像素的偏移量offset为（纵坐标 ×图片宽度+横坐标）×4。若扫描到的对应两个像素颜色不匹配，则生成一个错误像素及差别率加一，若两者之间有一张图片为空的图片则返回。最后得出的输出结果有两种：差别结果图，由所有错误像素组成的图片、差别率，通过与整幅图的像素面积取比率值。

由于analyseImages的计算方式也考虑背景的颜色，三维模型所转换的二维图像的像素数与背景的像素数之差相当大，导致差别率出现相反的结果。于是在实现系统中调用resemble得出的差别率可作为相似度来决定验证是否成功。analyseImages的函数定义如下代码所示。

表5-12二维图像相似度比较的调回函数处理

**function** analyseImages(img1, img2, width, height) {

**var** hiddenCanvas = document.createElement('canvas');

**var** data1 = img1.data;

**var** data2 = img2.data;

    hiddenCanvas.width = width;

    hiddenCanvas.height = height;

**var** context = hiddenCanvas.getContext('2d');

**var** imgd = context.createImageData(width, height);

**var** targetPix = imgd.data;

**var** mismatchCount = 0;

**var** time = Date.now();

**var** skip;

**if** (!!largeImageThreshold && ignoreAntialiasing && (width > largeImageThreshold || height > largeImageThreshold)) {

        skip = 6;

    }

// 根据图像的大小尺寸遍历所有像素点扫描一遍

    loop(height, width, **function**(verticalPos, horizontalPos) {

//若图像尺寸过大的时候就停止操作

**if** (skip) { // only skip if the image isn't small

**if** (verticalPos % skip === 0 || horizontalPos % skip === 0) {

**return**;

            }

        }

  // 获取两张图像的每个像素点的信息

**var** offset = (verticalPos \* width + horizontalPos) \* 4;

**var** pixel1 = getPixelInfo(data1, offset, 1);

**var** pixel2 = getPixelInfo(data2, offset, 2);

**if** (pixel1 === **null** || pixel2 === **null**) {

**return**;

        }

   // 忽略两张图像的背景颜色

**if** (ignoreColors) {

            addBrightnessInfo(pixel1);

            addBrightnessInfo(pixel2);

 // 如果两个图像的像素点具有共同的Brightness亮度值

**if** (isPixelBrightnessSimilar(pixel1, pixel2)) {

                copyGrayScalePixel(targetPix, offset, pixel2);

            } **else** {

                errorPixel(targetPix, offset, pixel1, pixel2);

                mismatchCount++;

            }

**return**;

        }

// 如果两个图像的像素点具有共同的RGB颜色值

**if** (isRGBSimilar(pixel1, pixel2)) {

            copyPixel(targetPix, offset, pixel1, pixel2);

        }

      });

从上述所描述的三维模型所转换的二维图像相似度的比较算法后。通过将两张图片的所有像素点进行遍历，并且获取到每个像素点所提供的像素点信息（如RGB颜色，像素点亮度值Brightness）。从而可得到对应两张二维图像的相差结果为mismatch Count值。在遍历二维图像的像素点过程中。如果程序遇到其中两张图片具有为空的某个像素点，则会跳过该像素点而不去进行获取该像素点的信息。为了提高像素点的比较遍历速度从而提高相似度的比较算法的性能。

因为在很多情况下，三维模型所转换的二维图像会有不相同的背景颜色。导致如果正常使用普通的比较算法将两张图像进行比较相似的时会有产生一定的偏差。导致比较结果不为实际。所以在该相似度的比较算法程序中，该算法有提供忽略颜色的相似度比较。当用户选择了忽略颜色的比较方法时，该算法会自动为每张图像当前的像素点进行添加Brightness亮度值（该亮度值可从HTML Canvas获得的数据）。然后将两个亮度值互相比较。若比较结果相同的时候意味着在当前两个像素点有相同的亮度。则在输出比较结果图片中产生一个灰色点Grayscale Pixel。为了方便用户可看到两张图片的相同之处。反之若亮度不相同时将相似度比较结果的相差值mismatch count加一。意味着当前两个像素点的特征不为相同。这样一来可以提高该相似度比较算法的准确度。在实际使用情况下更能提高实用性。

表5-3计算两张图片的相似度并输出比较结果

// 计算两张图片的相似度的相差结果

data.misMatchPercentage = (mismatchCount / (height \* width) \* 100).toFixed(2);

// 计算两张图片比较过程所花费时间 (比较算法性能)

 data.analysisTime = Date.now() - time;

    data.getImageDataUrl = **function**(text) {

**var** barHeight = 0;

**if** (text) {

            barHeight = addLabel(text, context, hiddenCanvas);

        }

        context.putImageData(imgd, 0, barHeight);

// 计算两张图片的相似度的相差结果

**return** hiddenCanvas.toDataURL("image/png");

## 本章小结

本章具体介绍面向 WebGL的基于自动语义标注系统实现和验证。本章首先给出原型系统的总体架构，随后对其中的部分模块的实现方式进行阐述。本章主要对基于三维模型的自动语义标注的两种标注方法进行详细的分析以及系统设计的阐述。随后基于该自动语义标注方法，扩展从三维模型到视频资源的自动语义标注的初步设计与实现。而且通过系统的验证后可得到相应的三维模型语义标注以及视频资源的语义标注。本章最后将本文所介绍的系统与其他相关的研究就在个方面进行了比较和讨论。

# 系统应用验证与实验结果

## 研究结果和意义

### 自动识别图像的语义标注方法

表5-4获取三维模型流程所需时间

|  |  |  |  |
| --- | --- | --- | --- |
| **Times** | **Get semantic** | **Get similar image** | **Unit** |
| 1 | 901 | 286 | ms |
| 2 | 849 | 346 | ms |
| 3 | 1430 | 327 | ms |
| 4 | 1480 | 302 | ms |
| 5 | 2230 | 318 | ms |
| 6 | 1790 | 322 | ms |

图1-4获取三维模型流程所需时间统计图

Fig.1-11 Three-dimensional model of semantic annotation process time graph

表5-5 三维模型语义标注获取数量

|  |  |  |
| --- | --- | --- |
| Times | Semantic amount per times | Unit |
| 1 | 10 | Tags |
| 2 | 9 | Tags |
| 3 | 8 | Tags |
| 4 | 9 | Tags |
| 5 | 11 | Tags |
| 6 | 10 | Tags |
| 7 | 8 | Tags |
| 8 | 7 | Tags |
| 9 | 9 | Tags |
| 10 | 6 | Tags |

图1-5三维模型语义标注获取数量统计图

Fig.1-12 obtain the number of three-dimensional model semantic annotation chart

### 图像的相似度比较的语义标注方法

## 系统的不足之处

## 本章结论

# 总结与展望



## 主要的研究工作总结

本文提出了一套基于本体面向 X3D 三维场景文档的半自动语义标注框架，旨在为三维文档的快速语义标注提供一个可行的解决方案。本文提出的框架利用本体作为语义标注的知识表现载体，利用规则定义来实现软件执行的自动化。本文的主要工作包括以下几点：

其一，提出基于本体面向 X3D 文档的半自动语义标注框架的总体架构，定义该框架与外界的交互行为以及其内部的主要功能模块；确定一个典型的标注流程在该框架中执行的主要步骤。

其二，对基于三维文档的语义标注进行本体建模。提出用于描述三维场景建模中基础要素的通用本体概念，给出通用本体需要满足的条件以及一个典型的通用本体实现。提出用于标注文档结构的文档本体和实现。提出利用通用本体进行领域本体的建模。

其三，提出从三维场景文档中抽取可视化信息的算法。这包括，识别出三维文档中所包含的可视化信息的种类；为每一种可视化概念类提供抽取算法；为每一种属性建立计算方法；为每一种可合并的属性建立合并的算法；为牵涉到多个物体的空间属性提供计算公式。

其四，建立根据用户制定的标注规则执行自动标注的标注引擎。这个过程首先包括提出用于描述自动标注行为的标注规则；给出标注规则的一般语法，规则定义中可以用到的标识符种类；定义用简单规则语法创建复杂的规则描述的方法。在此基础上，给出了标注规则一致性检测的算法，分别针对单个标注内部的一致性以及多个标注循环触发的可能性给出检测策略。随后，提出能够在大量三维场景文档中迅速找出符合条件的标注规则和场景节点，并给出从标注规则文本到该数据结构的构造方式，以及如何将抽取出的可视化信息分配到该数据结构中启动标注引擎的算法。

最后，建立原型系统。将本文提出的思想及算法实现为一个可运行的软件系统。该系统作为一个标注系统，包含以下若干功能：导入 X3D 场景文件并从中抽取出可视化信息；导入通用本体，文档本体以及领域本体，指导用户编写标注规则；显示 X3D 文件渲染场景及源代码，辅助用户进行部分标注的完成；导入待标注三维文档，各类本体以及部分标注，进行自动标注，并将标注内容进行持久化。作为系统的验证，选取了办公楼建模作为应用场景，建立了三维场景，标注规则，领域本体以及部分标注，验证了由标注引擎所自动生成的标注内容的正确性。

## 未来的工作展望

面向X3D基于本体的半自动化语义标注方法，还有如下问题有待进一步的研究：

第一，在本文现有的研究中，物体的形状信息难以计算。实际上，较多的标注行为都可以以物体的形状进行描述，在未来的研究中需要将物体的形状抽取融入到可视化信息抽取过程中去。

第二，面对大量规则或场景文档时，本文提出的规则匹配和应用算法仍然是不够的，因为需要建立庞大的条件网络图。在未来的研究中，将着眼于如何在该算法上进行优化。

第三，本文提出的方法仍然处于原型系统的阶段，面对企业级的使用仍然有诸多可扩展性和健壮性的问题需要考虑。

参考文献

1. Evans A, et al. 3D graphics on the web: A survey. Computer Graph (2014), http://dx.doi.org/10.1016/j. cag.2014.02.002.
2. Zhoufan Zhou, Hisao Utsumi, and Yuzuru Tanaka: X3D-Based Web 3D Resources Integration and Reediting, AMT 2009, pp 454-466.
3. WebGL Wikipedia, http://en.wikipedia.org/wiki/WebGL.
4. Nakhoon Back, A Standalone WebGL Supporting Architecture, WASET 2013, August 14 - 15, 2013, Venice, Italy.
5. Ben Houston, Wayne Larsen, Bryan Larsen, Jack Caron, Nima Nikfetrat, Clara.io: Full-Featured 3D Content Creation for the Web and Cloud Era, SIGGRAPH 2013, July 21 – 25, 2013, Anaheim, California.
6. Kouroupetroglou, G., Pino, A., Balmpakakis, A., Chalastanis, D., Golematis V., Ioannou N., Koutsoumpas, I. Using Wiimote for 2D and 3D pointing tasks: gesture performance evaluation. Gesture Workshop , Springer, (2011), 13-23.
7. Silveira, W. G. Manipulation of 3D objects in collaborative environments Using the Kinect Device. Federal University of Uberlandia. (2009).
8. LE Potter, J Araullo, L Carter, The Leap Motion controller: a view on sign language, OzCHI’13, November 25 - 29 2013, Adelaide, Australia.
9. Leap Motion Wikipedia, http://en.wikipedia.org/wiki/Leap\_Motion.
10. Three.js Wikipedia, http://en.wikipedia.org/wiki/Threejs.
11. JC Coelho, FJ Verbeek, Pointing Task Evaluation of Leap Motion Controller in 3D Virtual Environment, Chi Sparks 2014, The Netherlands, April 3, 2014.
12. Javi Agenjo, Alun Evans, Josep Blat, WebGLStudio – a Pipeline for WebGL Scene Creation, Web3D 2013, June 20 – 22, 2013, San Sebastian, Spain, pp. 79-82.
13. Leap Motion developing guide, https://developer.leapmotion.com/documentation/JavaScript/devguide/Leap\_Overview.html.
14. Y.Furukawa, WEB-BASED CONTROL APPLICATION USING WEBSOCKET, ICALEPCS 2011, Grenoble, France.
15. Anastasia Ailamaki, Bruce Maggs, Todd Mowry, Christopher Olston, Anthony Tomasic, Scalable Query Result Caching for Web Applications, VLDB `08, August 24­30, 2008, Auckland, New Zealand.
16. Gurudatt Kulkarni, Rani Waghmare, Rajnikant Palwe, Vidya Waykule, Hemant Bankar, Kundlik Koli, Cloud Storage Architecture, 2012 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA).
17. Kunert, A., Huckauf, A., Froehlich B. A comparison of tracking- and controller-based input for complex bimanual interaction in virtual environments. B. Froehlich, R. Blach, and R. van Liere, editors, EG IPTEGVE 2007,(2007), 4352.
18. Attene,  M.,  Robbiano,  F.,  Spagnuolo,  M.,  &  Falcidieno,  B.  (2009).  Characterization  of3D  shape  parts  for  semantic  annotation.  Computer-Aided  Design,  41(10),  756-763.
19. Bloehdorn,  S.,  Petridis,  K.,  Saathoff,  C.,  Simou,  N.,  Tzouvaras,  V.,  Avrithis,  Y.,Handschuh,  S.,  Kompatsiaris,  Y.,  Staab,  S.,  Strintzis,  M.  (2005).  Semantic  annotation  ofimages  and  videos  for  multimedia  analysis.  The  semantic  web:  research  and  applications, 592-607.
20. Kiryakov, A., Popov, B., Terziev, I., Manov, D., & Ognyanoff, D. (2004). Semantic annotation, indexing, and retrieval. Web Semantics: Science, Services and Agents on the World Wide Web, 2(1), 49-79.
21. Assfalg, J., Bertini, M., Colombo, C., Del Bimbo, A., & Nunziati, W. (2003). Semantic annotation of soccer videos: automatic highlights identification. Computer Vision and Image Understanding, 92(2), 285-305.
22. Bloehdorn, S., Petridis, K., Saathoff, C., Simou, N., Tzouvaras, V., Avrithis, Y., Handschuh, S., Kompatsiaris, Y., Staab, S., Strintzis, M. (2005). Semantic annotation of images and videos for multimedia analysis. The semantic web: research and applications, 592-607.
23. 潘翔, 张三元, & 叶修梓. (2009). 三维模型语义检索研究进展 [J]. 计算机学报, 32(6), 1069-1079.
24. 杨育彬, 林珲, & 朱庆. (2004). 基于内容的三维模型检索综述. 计算机学报, 27(010), 1297-1310.
25. Attene, M., Robbiano, F., Spagnuolo, M., & Falcidieno, B. (2009). Characterization of 3D shape parts for semantic annotation. Computer-Aided Design, 41(10), 756-763.
26. 王新颖, 王生生, 吕天阳, 庞纪明, & 王钲旋. (2007). Onto3D——一种基于本体的三维模型检索系统. 计算机研究与发展, 2.
27. X3D Specs. http://www.web3d.org/x3d/specifications/x3d\_specification.html
28. 林冬梅. (2004). X3D: 新一代 WEB 交互式三维图形规范. 计算机系统应用, 10, 1038-1043.
29. 曹雪. (2011). 面向网络协同的语义驱动的三维场景设计复用研究 (Master's thesis, 上海交通大学).
30. 施梦炜. (2011). 基于本体面向三维场景的语义标注研究与应用 (Master's thesis, 上海交通大学).
31. Josefsson, Simon. "The base16, base32, and base64 data encodings." (2006).
32. Wikipedia contributors. "Base64." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 3 Jun. 2014. Web. 5 Jun. 2014.
33. [24] Pittarello, F., & De Faveri, A. (2006, April). Semantic description of 3D environments: a proposal based on web standards. In Proceedings of the eleventh international conference on 3D web technology (pp. 85-95). ACM.
34. [25] Pittarello, F., & Gatto, I. (2012, May). A visual interface for querying ontologically and socially annotated 3D worlds for the web. In Proceedings of the International Working Conference on Advanced Visual Interfaces (pp. 377-381). ACM.
35. [26] Pittarello, F., & Gatto, I. (2012, May). An integrated ecosystem of interfaces for annotating, querying and browsing networks of Web3D worlds. In Proceedings of the International Working Conference on Advanced Visual Interfaces (pp. 794-795). ACM.
36. [27] Pittarello, F. (2009). An approach for tagging 3D worlds for the net. CELENTANO A.; YOSHITAKA A.; MARESCA P.; JIN Q, 88-93.
37. 张金钊, 张金锐, & 张金镝. (2007). X3D 虚拟现实设计: 第二代立体网络程序语言. 电子工业出版社.
38. 张金钊, 张金锐, & 张金镝. (2008). X3D 立体动画与游戏设计: 第二代立体网络程序设计实例. 电子工业出版社.
39. ZHANG, Y., & XIA, Q. (2006). JAVA Rules Engines Based on Rete Algorithm. Science Technology and Engineering, 11, 021.
40. xj3d. http://www.xj3d.org/
41. 李贞培, 李平, & 吴明. (2010). X3D 虚拟场景与 Java 交互研究. 计算机工程与应用, 46(016), 67-70.
42. 陈哲扬
43. Gstreamer开源多媒体框架 <http://baike.baidu.com/view/3468889.htm>
44. Yii Framework <http://baike.baidu.com/view/351219.htm?fr=aladdin>
45. MongoDB 数据库 <http://baike.baidu.com/subview/3385614/9338179.htm>
46. PHP Language <http://zh.wikipedia.org/wiki/PHP>
47. shimengwei

# 谢辞

本论文的完成，得益于导师蔡鸿明老师从选题的确定、论文资料的收集、论文框架的确定、开题报告准备及论文初稿与定稿中对字句的斟酌倾注的大量心血。蔡老师在毕业设计项目中给我创造了非常好的研究条件和设计环境。在整个项目过程中，蔡老师对项目的方向和进度给予了极大的关注，并对项目中遇到的问题给予很大的帮助。在此对尊敬的蔡鸿明老师表示真挚的感谢，感谢您在整个毕业设计的过程中给予的极大帮助。

本次项目的完成还得益于项目的指导老师姜丽红老师在百忙之中来到学校从项目的思路、项目的定位、项目的技术路线、项目的架构到项目的细节设计实现给予极其关键的指导和帮助。感谢蔡鸿明老师和姜丽红老师在毕业设计项目中传授的知识，给予的启迪和极大的帮助。

其次，我要感谢研究生阶段在生活上学习上帮助过我的同学和朋友们。这其中，尤其要感谢实验室里面的两位谢诚，黄承曦两位博士生，每次与你们讨论问题都能得到许多启发；我还要感谢施梦炜，汪涵等两位可视化小组的成员，没有你们打下的基础，将不会有本文的研究成果；感谢朋友非凡，在我遇到逆境和不顺时开导并鼓励我。

再次，我要感谢软件学院为学生提供的硬件设施，帮助我们在优良的实验室环境中学习和生活；感谢上海交通大学提供的优秀的网络资源和完善的学术资源，保障我的研究顺利进行。

最后，我要感谢一直以来默默支持着我的家人们。近三年来，无论我遇到什么样的挫折和失败，你们都会在第一时间关心我，为我出谋划策。感谢父亲在健康调理上对我的帮助，感谢母亲在生活自理上对我的关心。

如果没有大家的帮助本人是不能这么顺利地完成自己的毕业设计论文。我再次真心的感谢各位尊敬的老师、学长、学姐及同学在本人做毕业设计这段时间内的无言帮助与支持。谢谢你们！

# 攻读学位期间发表的学术论文目录

1. Do Nhat Quang, Hongming Cai, Lihong Jiang， Leap Studio - A Virtual Interactive 3D Modeling Application based on WebGL. In 2014 International Conference on Digital Home (ICDH 2014)