计算机科学 1996 Vol. 23 No. 2

可视计算机组成技术

周 璐 蔡 勋 李晓梅
(国防科技大学计算机系 长沙410073)

摘 要 可视计算强调对感兴趣的信息的可视描述和可视信息的直接处理，近年来它已成为一种重要的计算风格。本文我们对提供可视计算平台的三种主要技术作了研究和讨论。这些技术反映了可视计算机的不同用户——程序员、最终用户、及科学工作者的不同要求。我们首先讨论可视程序设计的重要发展，然后对可视界面及可视化进行讨论，最后概述可视计算的现状，并指出在将来的工作中可视计算应该强调的关键研究领域。

关键词 可视程序设计，程序可视化，科学可视化，直接操作，虚拟现实。

可视计算是在现代微处理器、计算机图形学、多媒体等技术日益发展成熟的条件下，新兴的一个研究把计算机构造成为可视建模设备的活跃领域。可视计算强调把抽象的事物以可视的形式反映给用户，是使计算变得更容易，扩大人们的认识范围的一个有力手段。本文探讨了可视计算的当前工作及它对将来可视计算机发展可能产生的影响。

1. 可视计算的分类

不同的使用者对计算机有着不同的要求，终端用户关心的是用户界面；程序员关心的是程序可视化和可视程序设计；科学工作者和数据分析员关心的是可视化和模拟。从这一角度来看，可视计算可分三个主要的分支，即：程序设计可视辅助、可视界面及可视化，如图1。

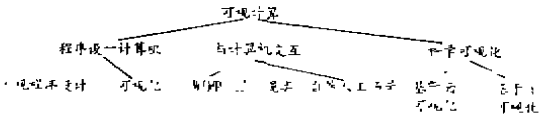

图1 可视计算分类

程序可视化和可视程序设计是提高程序员生产率和正确性的有效途径。前者主要用于程序构造，后者主要用于程序调试、程序行为理解以及数据可视化。第二个分支是与计算机的可视交互。依据实现交互的技术，可将各种系统分为三类：WIMP（Windows 窗口，Icon 图符，Menu 菜单，Pointing 指向），虚拟现实(Virtual Reality)，及自然人工景物(Natural Artifact)技术。第三个分支是科学可视化，它是可视计算中发展最为迅速的一个领域。它的迅速发展部分归功于计算能力和三维图形的发展，因为直到最近，才有可能绘制出一个用户可与之交互的三维环境的令人信服的界面。当前有两种主要的科学可视化方法：基于面的可视化方法和基于体的可视化方法。

下面我们基于上述分类进行详细讨论，并介绍各类中的典型系统。

2. 程序设计的可视辅助

图形有强有力的表达能力，而传统语言又不尽人意，人们便对使用图形作为程序设计的辅助手段产生了极大兴趣。传统程序设计语言的主要缺陷在于它的线性程序设计风格。图形增加了表达的维数，使程序员能直接把思想中的概念反映到计算机中，这种程序设计风格对人来说更为自然。由于它自动地完成程序设计中的琐碎工作，因此增强了程序员开发和调试代码的能力，在程序设计的简单性和高效性上取得重大突破。

可视程序设计系统使程序员可以通过二维或高维计算功能和单元的显示技术来合成程序；程序可视化则是通过提供对程序不同方面的特征的可视来帮助人们调试和理解程序。它们的进一步分类见图2。可视程序设计系统根据图形在构造程序中的使用方式来进行分类；第一类是图形界面系统，用户通过与系统交互来指导系统生成程序；第二类是可视语言系统，除了用图象代替正文之外，这类系统与传统程序设计系统类似，根据用来生成程序的图象不同，对可视语言系统可作进一步的分类。

可视化系统是根据它们是否显示程序代码、程序数据或基本算法来进行分类的。根据它们是静态还是动态，程序代码和数据可视化系统还可进一步划分。

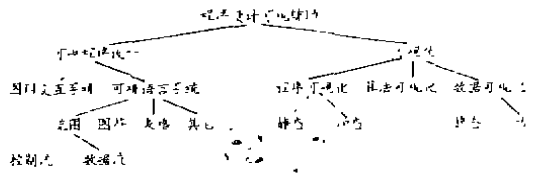

图2 可视程序设计分类

2.1 可视程序设计系统

·图形交互程序设计(Programing through graphical interaction) 图形交互系统提供给程序员交互式图形界面，程序员通过交互来指导系统生成一个供最终用户使用的程序，这种程序设计也叫不可见编程。大多数这类系统是解释执行的。

图形交互系统大致可分为两类：带推理机制的和不带推理机制的系统。前者的系统根据程序员提供的示例来确定程序中使用的循环、条件、变量及常量，在诸如用户界面这样的良定义的领域中，用带推理机制的图形界面进行程序设计已取得了成功。如界面管理系统 Druid 和图形编辑器 Mondrian。不带推理机制的系统则要求程序员在示例中对控制加以明确的说明，多数系统要求程序员完成一定的步骤，系统记住这些步骤，并把它们翻译为程序。典型的系统有 Pygmaliont Graphical ThingLab。

·可视程序设计语言系统 可视程序设计语言有一个图形符号集，这些符号按一定语法规则排列就构造出程序，图形符号的排列有严格定义的语义。就此而言，传统程序设计语言和可视程序设计语言的唯一区别，在于后者使用的是图形符号，而不是正文表达式。但在编程的简单性和效率方面，二者相去甚远。根据在创建程序时使用的图形抽象不同，现有的大多数可视程序设计语言可划分为三类：流图、图符、表和表格。

流图(Flow diagram) 基于流图的可视程序设计系统使程序员能使用各种类型的图表和图来构造程序。流图被用来说明程序的数据流或控制流。程序员创建的图形说明被编译为某一传统程序设计语言，或直接由系统进行翻译。Grail 系统把用户描述的流程图直接翻译成机器语言程序；PASCAL/HSD 则是在 PASCAL 中加上标准流程图的图形符号，还有一些用数据流图来支持数据流语言程序自动生成的系统，如 PROGRAPH。

图符(Icon) 在图符系统中，程序是通过称为图符的专门设计的图形符号及其之间的联系来构造程序的。通常，图符的互联方式是由系统对图符的排列赋予一定的语义来说明的。多数基于图符的程序设计系统提供预定义的图符，并允许用户自定义图符。程序员通过某种方式把它们连成需要的控制流的路径表示来构造程序。使用图符的程序设计系统有：SunPict，Show and Tell，VenLisp 等。

表/表格(Tables/Forms) 这类系统允许用户通过使用表或填写表格来构造程序，表格填写被认为是一种与用户交互的容易方法，因而在许多系统中被大量地使用。如 QBE(Query By Example)允许用户用二维表说明对关系数据库的查询；FORMANAGER 使用户能用表格开发事务信息系统；FORMAL 是一种基于表的数据库语言。

其它一些系统综合使用了上面讨论的各种技术，如 Vu。它是一个开发图形用户界面的可视环境，为用户提供许多标准用户界面对象，用户对这些对象的使用是以高度交互的图形方式完成的。Vu 系统使用直接操作技术，如橡皮带式生成线(rubberbanding)、拖动(dragging)、及填表来实现对象的用户化。另一个使用类似技术的用户界面开发系统是 DialogEditor。

2.2 可视化系统

·程序可视化 程序可视化系统提供程序代码的静态或动态显示，帮助对程序代码结构和程序控制流的理解，以及对程序性能调试的分析。

静态程序可视化系统通常借助于流程图来描述程序数据流(如 Visual Sytax Editor)，或借助整齐打印的空行、空格、缩进格式、和注释等(如 SEE)来增加程序的可读性。动态程序可视化系统以动画形式显示程序的执行过程，或以对应于程序的动画图画，或以加亮程序代码的方式，显示程序运行时的控制、程序、或数据视图。动态程序可视化对调试和分析分布式及多处理机软件特别有用，使程序员能了解系统与正在执行的程序之间的相互作用。PECAN 是典型的多视图动态可视化系统。

·算法可视化 这种系统显示程序的基本操作，不仅体现数据的转换和访问，也体现控制流。Ron Baecker 的系统能生成帧序列，产生电影效果的算法动画显示；BALSA 系统则是在算法代码中插入有趣事件(interesting event)过程调用来生成动画。

·数据可视化 数据可视化系统以静态或动画的方式来显示程序数据。静态数据可视化系统通常用于数据结构的自动生成，系统除提供显示库外，还允许用户自定义显示方式，并支持各种数据结构的可视、编辑、和浏览等功能，如 INSENCE。动态数据可视化通过交互式图形显示来定义程序数据结构之间的关系，描述程序运行过程中对数据的访问和修改。这类系统已在并行和分布式程序的性能调试中广泛使用，如 PV。

3. 最终用户与计算机的交互

八十年代以来，使用的方便性已成为决定一个计算系统能否取得商业成功的主要因素。结果，大多数现代计算机都配备了可视的交互式直接操作界面。

用户界面的发展导致了软件和硬件上新技术的进步，以及新的交互模型的开发。近年来，界面已从单纯的正文方式发展到图形方式和使用象头盔显示器及数据手套这样的高级界面设备的直接操作方式。

对最终用户与计算机的交互的分类可分为两级，如图3。第一级基于实现交互所用的软、硬件技术。我们讨论三个主要的技术：WIMP、VR、人工自然场景。使用上述分类中的每种技术也还有不同的交互风格，这些交互模型或隐喻构成了分类的第二级。

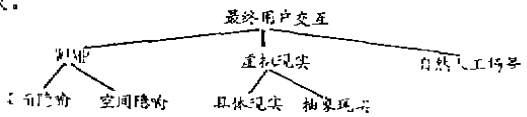

图3 最终用户交互分类

3.1 WIMP 界面

现有的大多数可视界面是以窗口(Window)、图符(Icon)、菜单(Menu)、指向(Pointing)为特点的。因此称为 WIMP 界面。几乎所有的工作站和个人机都配有 WIMP 技术。在过去的十年中，基于 WIMP 技术的可视界面成功在推动了非计算机专家对计算机的使用。WIMP 技术已被用来实现几种信息组织和交互的隐喻，重要的有桌面和空间隐喻。

·**桌面隐喻**(Desktop metaphor) 桌面隐喻在一个计算机系统中提供常见的事物对象和操作，来模拟一个类属事物。它利用用户对常见事物的熟悉，构造一个友好的交互环境。现有的大多数可视界面用 WIMP 技术为用户提供桌面隐喻。当前 WIMP 界面的一个重要特点是用与对象-操作相结合的点取(point-and shoot)方式在屏幕上移动对象，并实现一定的操作。例如，要打印一个文件，用户先通过指向它来选取该文件，然后把它拖到打印机图符上，同样，在一个文件图符上连按两下鼠标按纽，不仅选择了该文件，同时也激活了与之联系的应用程序。

最早用桌面隐喻实现用户界面的是 Xerox PARC 开发的 Xerox Star 计算机，其后是 Apple Lisa 计算机。然而 WIMP 风格界面及桌面隐喻的流行应归功于1984年推出的 Apple Macintosh 计算机。自 Macintosh 出现以来，WIMP 技术已成为工作站和个人机的一个标准部分。

基于桌面模型的界面已在办公室应用中取得了巨大成功，如文档准备，文件和文件夹管理，以及电子邮件等。然而，它们在处理专门化的应用时，能力非常有限，如3-D CAD/CAM，科学可视化，以及模拟等方面的应用。这一限制促使人们开发新的界面模型和技术。

·**空间隐喻**(Spatial Metaphor) 空间隐喻的交互包括移动和操作物理模型中的对象。Boxer 系统用空间框隐喻(spatial-box metaphor)表示许多包含语义概念。方框依层次关系嵌套，每个方框可以打开或缩小为符号，取决于用户在层次中的位置。另一个空间隐喻的系统是 Henderson 和 Card 开发的 ROOMS 窗口管理器。许多系统在空间隐喻的实现中使用了 WIMP 技术。

界面设计的另一方法是超文本结构(hypertext)，它的物理模型是空间网络，其中结点用来表示界面状态。由于在用户界面的状态转换表示和等价超文本之间的结构同性，该模型对界面设计者有极大的吸引力。对于超文本结构，基本的交互方式是导航(navigation)。在每一状态下，用户都能修改网络的对象和结构。超文本方法的优点在于它把设计者或工程师可能使用的界面映射为有限状态自动机模型。然而，在大的非结构化超文本中，这种方法容易出现迷路(disorientation)问题。

3.2 虚拟现实(Virtual reality)

计算的不断发展使得计算机成为人类环境中的一个不可缺少的部分，渗透到人们生活中的各个方面。现在，计算机对特定环境的模拟能力，更使它成为使用者能在其中进行研究和工作的环境——虚拟现实，它实际上是人们在真实世界中活动的一种隐喻。通过建立一个具有真实感的模型，并使用合适的输入设备，如感知手的位置和方向的数据手套和头盔显示器，使用户感觉自己确实处于计算机所模拟的这个环境之"内"，这是单靠观察绘制在外部屏幕上的图像无法做到的。

虚拟现实可以看作是由模型、成像部分、以及操作集三部分构成的。模型包括一个坐标空间及该坐标系统内用户与之交互的图像；模型中显示的图像

由界面的成像部分来实现，应当既能显示图像，也能在模型中移动图像；除此之外，还应提供用户与应用环境交互的操作集，它允许用户使用应用环境的功能，同时保证界面中创建的整个虚拟现实模型的一致性。虚拟现实可分为两类：基于物理现实的系统和能处理抽象概念的系统。

·**物理现实**(Physical reality)　虚拟现实系统中的对象在真实世界中有其对应物，系统模拟真实世界对象的物理(通常是视觉)和行为特征。如 North Caronina 大学的 WalkThrough 系统，它是一个建筑模拟系统，设计者可用头盔显示器和 Polhemus 3Space 磁跟踪器模拟在模型中走动的感觉，并获得有关建筑物的各种物理数据。其中的建筑物模型是用 AutoCAD 实体建模程序包来构造的。

·**抽象现实**(Abstract reality)　许多可视虚拟现实系统允许用户对抽象信息进行可视和操作，用户看见的信息在真实世界中没有物理(视觉)表示。例如压力和能量场，温度和地震测量中获得的数据。这些系统强调信息的表示方式应使用户容易地看见和操作对于他来说是重要的那些方面。以压力和能量场为例，由于它们分布在一个三维空间中，所以可用各种颜色信息来显示空间中能量或压力的分布。当信息与物理空间或时间的联系不十分清楚时，通常比较难以设计出对用户直观上清楚的表示方式。DataPaper 是一个抽象现实系统。

3.3　自然人工场景(Natural Artifacts)

自然人工场景代表了用自然方式(如手势、文字书写、及口述语言)与计算机交互的发展方向。

近年来，用笔输入与计算机交互的方式引起人们很大兴趣，这导致了称为基于笔计算(pen-based computing)的计算范型的出现。其思想是仅用笔来与计算机交互——不是作为对键盘和鼠标的扩充，而是取代它们。基于笔的计算产生了对识别技术的迫切要求，因为目前的技术尚不足以解决手写体正文识别问题，但神经网络技术已在该领域中显示出美好的前景。

另一重要发展是用声频命令来与计算机交互。声频界面可分为两类，一类是用户以口述形式向计算机发出命令；另一类是计算机以声频(包括语言声频和非语言声频)向用户提供反馈信息。

在以自然方式与计算机交互中，手势的使用越来越多。Rhyne 把手势作为"笔和纸的电子模拟"，应用于正文编辑，其界面是笔和图形输入板。Pausch 和 Williams 在他们的 Tailor 系统用数据手套识别在三维空间中手的运动产生的手势；Murakami 和 Taguchi 建立了一个基于神经网络的日本手势语言的识别系统。

多媒体在界面中的应用是交互技术的另一重要发展。界面已开始使用现场动作视频、交互式三维图像、动画和正文等媒体来生成具有真实感的易于使用的界面。计算机作为具有紧密结合的硬件(如 CD-ROM 驱动器和视盘机)的多媒体平台已进入市场。

4. 科学可视化

在过去的十年中，高速数字处理机的极大进步使科学家能够解决以往被认为是计算代价太大或不可计算的问题。然而，大多数这样的问题，其结果数据量极大，或极为复杂，以致于无法用传统技术来理解。科学可视化使科学家能把大体积数据转换为有意义的图形，图形提供更多的维数，用动态的图形来显示线性数据，从而使科学工作者能够了解数值解的全局特性或数据各组成部分之间的因果关系，观察数据随时间的变化；或通过显示交互来更好地考查数据，发现异常，找出计算错误。实际上，科学可视化延伸了人的视觉系统，不用这种技术，人就不可能观察到那些过于细小、过于复杂、或过于抽象的东西。

科学可视化在实现数据可视和交互的科学家界面时，利用了计算机图形学、图像理解与合成、用户界面、计算机视觉、及计算机辅助设计的技术，取得了极大成功。目前，科学可视化主要应用于计算科学和工程设计应用中产生的数值数据的可视。

直到最近，可视化软件工具一直是由图形子程序库和动画程序包组成的。图形子程序库是一个低级图形操作集合，用该库实现可视化。用户需要用传统语言编程，并在程序中嵌入对图形子程序的调用。这一过程不仅要花费许多时间，而且要求用户有特殊的技巧，例如，要了解图形库使用的数据结构、影响图像成像的观察和光照属性、以及库所使用的绘制过程。

使用动画程序包生成可视化软件时，用户首先(通常以批方式)生成可视数据，然后用包提供的功能来实现可视化。尽管与图形子程序库相比，使用程序包更为容易和高效，但也还有许多问题。如大多数

程序包的应用范围非常狭窄，数据输入格式也十分有限，常常需要将一个模块产生的输出数据转换为另一个模块能够接受的数据形式。

已有一些可视化程序包被开发出来以解决上述问题。这些包把生成可视化的整个过程分为若干步骤，并提供能简便地管理这些步骤的软件工具。例如AVS，它提供若干软件模块，可用交互式图形技术在流网络中进行连接，在许多情况下不需要进行传统程序设计就能生成一个完整的可视化应用程序。这种方法的一个重要好处是可以简单、快速地开发出可视化应用软件。apE 是 Ohio 超级计算机中心开发的一个类似于 AVS 的科学可视化软件包。它们在构造数据可视化时，都使用数据流分解和可视程序设计；但它们的组成、使用的数据元素以及可移动性不同。

当前多维数据可视化中主要使用的方法有两种：面可视化和体可视化。

4.1 基于面的可视化

按我们的分类，面可视化包括了线框可视化(wire-frame visualization)，因为二者除了图像产生的最后一步外，十分相似。在面可视化中，绘制之前先把结构的表面表示为一个表面元素集，可用于表面元素产生的方法很多。

基于表面的可视化对显示几何物体有效，特别是对于象工业设计和制造这样的应用领域，面可视化技术尤为有效。但对于象能量场和地震测量的数据显示，这一技术并不十分令人满意。部分原因是由于表面绘制要求画出每一结构特征的等值线，这是一个非常耗时、工作量很大的过程，而得到的多边形网格不仅可能漏掉有用的信息，而且会使人产生错觉，认为所描述的特征具有良定义的几何表面。

4.2 基于体的可视化

体可视化用于三维矢量场或标量场采样数据的可视，不需要对数据进行几何基元的拟合。这种方法把三维数据表示成类立方体的构件(称为体素(voxel)或体元(volume element))集合，通过计算每个体素的颜色值和部分不透明度值，然后合成所有投射到像平面上同一像素的体素对最后成像的贡献得到图像。有两种把体素投射为像素的方法：一种是光线投射(ray-casting)，使光线从前向后穿过体数据，累计每一遇到的体素的视觉特性；另一种方法称为合成(compositing)，体素由后向前投射到屏幕上，每一层的视觉特性与前面各层的混合。

体可视化与面可视化相比，其主要优点在于图像质量更高，无需精确定义表面的几何性质，以及能够描绘不确定边界的现象，如能量场和扩散分布。

5. 将来的研究

5.1 程序设计的可视辅助

可视程序设计和程序可视化正越来越受关注，过去的几年中已推出许多有趣的系统，而更多的正在开发。尽管已作了很多努力，充分开发图形在程序设计中的潜力还有待于作进一步的努力。

·可视程序设计　显示屏大小有限是可视程序设计系统中的一个主要问题。通常，程序的可视表示比正文表示占用更多的屏幕空间，这对大型程序而言是个严重问题——屏幕上只能放下很小的一部分程序。这个问题可通过屏幕滚动和缩放技术得到部分解决；另一可能的解决方法是设计以有限应用程序类为目标的系统，这样就有可能对这些应用程序类使用更高的抽象，以减少规范说明。

尽管通常图象能帮助开发和理解程序，但由于不同的用户可能给同一幅图象赋予不同意义，有时也会引起混乱。要准确地给出某种含义的可视表示并不是件很容易的事，对于应用范围相对较窄且问题集中的系统来说，这个问题比较容易解决——可在系统应用范围内使用严格定义的符号。

在可视程序设计系统的评价方面所做的工作很少。可以从可视语言和可用性角度加以评价，就语言级别、应用范围及可视表达能力对可视程序设计系统进行特征刻画，这能帮助理解可视程序设计系统的能力。Glinert 从可用性和软件工程角度对可视程序设计系统进行了讨论。

分布式和多处理机环境下软件的可视程序设计是一个较新的领域。软件模块交互的复杂性使得很难用传统的程序设计语言和技术开发这样的程序，可视程序设计则可清楚地显示程序模块之间的通讯和同步依赖关系，大大降低这种复杂性。

·程序可视化　在程序可视化系统中，一个主要的问题是大量数据结构和程序的显示。由于屏幕尺寸的限制，决定"显示什么"和"如何显示"就极为重要，很难动态地选出重要的部分，并给出具有视觉吸引力的有效显示。

对同一显示信息和动画显示不同的视图已被证

明是信息可视化的有效途径，但视图和动画的产生需要大量时间和计算。如何降低这一开销，使程序员能快速地生成多幅视图和动画，对它们加以比较，从中选出最好的，这已变得十分重要。另一问题是图形子程序与源代码的结合(binding)。为了显示程序代码和数据，需要在程序源代码的合适位置插入对图形子程序的调用，但这种方法并不受欢迎，因为必须对源代码进行修改和重新编译。有些系统允许程序员创建这种结合，但并不真的在程序源代码中插入图形子程序的调用。

用程序和数据可视化技术对分布式及多处理机软件中进行调试和分析的工作刚刚起步。已有一些系统允许程序员说明程序流，并用可视技术使程序执行和数据访问可视。然而为了使可视技术的潜力得到充分的发挥，还有更多的工作要做。

5.2 与计算机交互

·WIMP 界面　尽管在基于 WIMP 技术的界面设计中已取得成功，并有了大量的经验，低质量的 WIMP 界面仍在生产，主要的问题是，缺少开发良好界面的支持工具。用户界面管理系统(UIMS)的目标是通过提供界面开发的辅助工具来解决这一问题。过去十年中，在这一领域已有重大进展；并有一些小的 UIMS 商业产品出现。大多数这类工具都要求程序员非常了解人的因素，而实际情况并非如此。其结果是，尽管界面生成变得更快、更容易，但生成出来的界面并不一定更好。在界面的自动设计领域还需要进行研究，自动设计工具能在创建界面时考虑人的因素，保证最低质量能达到某一水平。这一方面的工作已经展开，且已开发出一些原型系统，如 UofA* 和 UIDE。

另一个研究领域是界面评价。目前这方面的方法和工具都严重不足，原因在于界面无法脱离使用者来进行评价。而使用者的熟练水平与期望水平的差别使得评价更为困难。一些初级界面评价工具和技术已被开发出来，但还有更多工作要做。

当前 WIMP 界面的发展要求数据和信息向高度空间化模型的方向发展，这将使物理虚拟现实中的工作能并行完成。然而，与正文或逻辑信息交互的问题和与数值数据或物理系统模型交互的问题是截然不同的。超文本是构造和联系不同种类的信息的一种方法，与超文本作为界面风格使用相关的许多问题来自于超文本本身的根本问题：在一个几乎没有结构信息或界标的松散结构超文本中应该如何导航？如何对一个大的超文本提供合适的入口点？应提供何种隐喻来支持与抽象信息的交互？有没有可能用超文本结构表来解决浏览时产生的迷路问题，尤其是当使用者对这些概念的解释各不相同的时候？

一句话，关于超文本，许多问题尚未解决，有一些刚开始被探讨，如超文本的哪些特性能加强它的可用性？大规模超文本的开发可能昂贵且费时，但正文到超文本的自动转换是当前研究的一个焦点。尽管存在这些问题，目前还是有许多精力花在超文本的研究上，因为这可能是最终处理大量不同类信息的唯一途径。在 WIMP 技术的每个主要成份(窗口、图符、菜单、指向)中也尚有工作可做，它们都是发现和创新层出不穷的活跃的研究领域。

·虚拟现实　在虚拟现实能被广泛应用之前，还有大量的研究工作要做，其中之一是确定虚拟现实界面的可能应用领域。当前的应用系统可分为以下几类：科学可视化和模拟，如流体力学模拟系统；游戏；数据测量，如 Walk-Through；复杂系统控制等。尽管提出了虚拟现实界面在心理学研究、医学研究、教育、及协同工作中的应用等一些新领域，但所做的工作并不多。多用户的虚拟现实可望用于设计、培训、及娱乐等领域。

许多现有的虚拟现实应用程序是用低级工具开发的，既费时又昂贵，这不利于对其它技术的探索，也限制了它的应用。因此需要开发支持生成这些界面的软件工具箱和 UIMS。首先要确定虚拟现实界面的一个公共需求集，并且要有支持这些需求的交互技术。这一领域正随着虚拟现实工具箱(如 Alberta 大学的 MR 和 VPL 的 RB2)和对话管理系统的出现而受到重视。

另一个非常活跃的研究领域是开发帮助建立虚拟环境的可视工具。开发由数以千计彼此联系并与用户交互的虚拟对象组成的复杂虚拟环境系统，是不可能用传统的实体建模方法做到的，因为需要说明的视觉及行为信息太多了。解决这个问题的方法之一是开发封装对象的视觉及交互行为的对象模型，这样就能在虚拟环境中对对象的属性加以说明，如3dm 中所做的。

随着虚拟环境变得越来越庞大，越来越复杂，并开始引入多于一个的参与者，分布式计算将起到重

要作用。一台计算机将难以满足下一代虚拟环境的计算和用户界面要求，因此需要开发帮助在计算机网络上建立和管理虚拟环境的工具和技术。许多研究人员已经开始了对虚拟现实中的计算问题方面的探讨。

虚拟现实中使用的昂贵设备阻碍了它的广泛使用和新的虚拟现实的开发。头盔、数据手套及合适的界面板加在一起高达上万美元。除此之外，这些设备也还存在着严重的人机工程学和技术上的限制，包括：头盔显示器的分辨率低、重量大，I/O 带宽低，位置传感器的发送方和接收方之间的串扰。目前正在尝试使用低造价、低分辨率、功能有限的设备进行虚拟环境的研究。

·自然人工场景　实现与计算机以人类日常交互的方式交互是有趣的，然而难以做到，因为这是一个相当广的领域，包括许多不同的技术，而这些技术的信噪比都很高，难以准确判断出真正的输入；而且声频、手势以及手写体的使用很可能是高度依赖于使用者的，要解决这个问题，要么使系统专用于特定的用户风格，要么使它一般化到能够独立于使用者。这两种方法都有各自的问题。此外，用这些技术进行交互可能比纯 WIMP 界面需要更多资源。微处理机技术和硬件设备的近期发展，已使建立一个用户能负担得起的界面成为可能，对在界面中使用自然人工场景的研究也因此得以加速发展。

5.3　科学可视化

在过去的两、三年中，研究人员在科学可视化领域中做了大量工作，除了一些研究原型外，还出现了象 AVS 和 SunVision 这样的商业产品。但可视化软件的生产仍是一个困难、费时的过程，因此需要一个软件环境，能使不是程序设计专家的科学工作者容易地生成新的可视化软件。可视程序设计技术在程序可视化中的应用是一个令人振奋的有益的研究领域。

科学可视化的算法和数据结构是一个重要研究领域，因为可视化生成通常是一个计算密集的过程，算法上一个相对较小的提高，也能使生成图像的总时间大大减少。这些算法可分为几类：加快可视化（体可视化和面可视化）的算法、由图像数据构造体的算法以及构造体数据几何模型的算法。

操作的交互性和简单性是科学可视化软件对专业人员能否接受的关键因素。使用者应能简单、快速地操作可视图像，以更好地理解数据。此外，可视化环境应该是可用户化的，这样用户就能对环境进行剪裁，适应自己的风格。到目前为止，科学可视化设计者们对计算问题一直比较关注，而对专业人员的用户界面需求则很少注意。现在在计算领域已取得了显著进步，应该有更多的精力投入到对用户界面的研究上来。

可视化软件的并行技术正在得到越来越多的重视。可视化软件，尤其是交互式可视化软件的开发，需要极大的计算能力，多处理机的使用看来是提供这种能力的一个可行的办法，基于网络的可视化将是解决科学可视化的计算和交互的需求的关键。

本文我们对可视计算的三个主要分支进行了讨论，这些研究的综合将导致可视计算机的出现，它将是一个使我们能够直接观察世界，并与之交互的工具。虽然未来技术的发展还很难预测，但我们相信，可视计算机需要的基本元素都已具备了。

与可视计算机交互需要在强大的功能和使用的方便性之间取得平衡。在向更高级界面发展时，保留桌面隐喻的简单性是非常重要的。下一代可视计算机的设计者遇到的挑战是：如何用功能更强的计算机、输入/输出设备、以及复杂的软件来减轻使用者的认识负担，而不是用稀奇古怪的设备和显示使他们眼花缭乱。

参 考 文 献

[1]Ginminder Singh, Mark, H. Chignell, Components of visual computer: a review of relevant technologies, Visual Computer.9(1992)

[2]Bricken, M., Visual reality learning environment: potentials and challenges, Computer Graphics, Vol.25, Aug. 1991

[3]Livien, R., Visual programing, BYTE, Feb. 1986

[4]Nielson, G., Visualization in scientific and engineering computation, IEEE Computer Vol. 24, Sep. 1991