Generalized Fisheye Views

泛化鱼眼视图

摘要  
在很多情境下，人类经常把临近的事物展示得非常细致，但是较远的事物就只描绘出主要轮廓。这种行为其实说明了类似的鱼眼视图可以被计算机用于展示大量信息结构，比如程序、数据库、在线文本等等。本文探究鱼眼视图展示效果，依次讨论自然界中的类比，泛型形式，继而研究一个特殊的实例，一个计算机程序以及展示示例和评估。

介绍

计算机程序代码，数据库，结构化表格，在线文本，系统菜单和地图——用户被强制在一个小至24x80的显示器展示的窗口中浏览所有这些有潜在可能非常巨大的结构。问题是太多的信息需要展示出来了，从局部细节到全局的结构。目前通用的展示界面就是供人将目光聚集到结构中的一点的一个小窗口。比如，一个简单的编辑器窗口可能展示程序代码中的一行或者和它连续的几行。一个系统菜单可能展示当前可选项的集合。用户通过滚动或旋转窗口页面来把显示导向自己希望的部分，这很容易迷失方向，也就是说跳转到了一个错误的地方但不知道该怎样找回到正确的目的地。这大概是由于这样的视图缺少全局结构的信息，尤其是当前显示的内容在全局结构的什么位置。为了解决这样的问题，有一些技术被发明出来。最有名的类比就是可变焦距镜头——使得全局结构和局部细节都可以展示出来，或者是并排展示，或者是在一个序列内展示（最早的例子是Englebart的增智研究中心）。

我们探索了迥然不同的一种视图策略，它就像广角镜头，或者说是鱼眼镜头一个性质。鱼眼镜头可以在将近处展示得非常详细的情况下依然完整展示整个画面——只是缩略较远处的细节。一个类似的说明性的漫画叫做“纽约人眼中的美国”，这是一张来自Steinberg的海报，现在也被很多其他城市模仿。在这张海报中，曼哈顿中城的每条街道都画了出来。向西看，新泽西只是在一条标注着“哈德孙”字样的灰蓝色带子另一边的一块色斑。这个国家剩余的部分略缩成了一些主要地标，比如芝加哥，落基山，加利福尼亚等等。这确实是一种扭曲的视角，它有保留了美国在纽约人眼中的重要特征，省略了其余所有的细节。这幅图不仅可以让纽约人回答诸如“最近的邮箱在什么地方？”这种问题，也可以回答范围更大的问题比如“去落基山滑雪，从芝加哥还是从洛杉矶转机方便？”如果纽约人的鱼眼视角可以让他们回答这类问题，那么这种类比的视图也会在计算机界面上大有所为。

鱼眼视图最根本的动机是在局部细节和全局关系上取得一个平衡。局部细节是为了能在局部进行交互，正如在中城找到最近的邮箱或在一篇文章中编辑特定的一行。全局环境是为了告诉用户还有哪些部分也一样在大范围内和他们的大致位置（比如，落基山在西部，比芝加哥远但是比洛杉矶近）。全局的信息在仅仅展示局部细节的时候也是重要的（比如，C语言中的else语句实际上依赖于可能在较远处的if语句）

通过分析纽约人奇特视角的例子，也就是细节和距离的权衡，可以很容易想象鱼眼视图应用在很多领域的例子。本文先列举一些自然情况下鱼眼视图的例子，进而回到鱼眼视图在计算机交互界面的应用。

自然情况下的鱼眼视图

我们着手研究自然情况下的鱼眼视图有以下几个原因：一方面，作为认知心理学家，我们对人类如何在大脑中表示大型结构感兴趣。更相关的是，我们认为如果鱼眼视图是无处不在的，原因可能是它们自然地存在人类交互活动中，进而产生效果更好的交互界面。另外，我们希望了解更多和这种视图形态有关的东西，期望获得的结果可以有助于鱼眼视图界面的设计。

我们用简单的范例做了一些实验。实验对象被要求去想象一个新搬来的邻家小孩，这个小孩被告知了一些信息（包括州，总统，历史事件等）。实验对象的任务是列举10种他认为小孩可能知道的信息。经验上的鱼眼构想应该是，小孩了解的要么是非常重要的信息，要么是离家近的信息。这样的鱼眼子集确实也被人列出来了。按州划分，在新泽西和德克萨斯的实验对象提到了很重要的州（纽约州，加利福尼亚），但是出现了地理位置的区别（德克萨斯人列举了阿肯色，新泽西人则列举了康涅狄格）。类似的，实验对象列出了非常杰出的总统（华盛顿，林肯）和最近的总统（卡特，里根）。

通过其他技术，我们发现在一家大公司内的人们了解的也是管理层的鱼眼子集。雇员只是知道本部门的本地领导，但是其他部门的只知道副主席这一级别的。

我们也研究了科学家对于学术界的视角，发现和个人研究相近的领域被放大，一个实验心理学家会把管理和市场看作更相近的理论，而心理学和精神病学就分别的很清楚。对于经济类学校的人来说他会得到完全相反的结论。

通过分析三大地区的12种报纸上报导的类型，我们发现新闻编辑也形成了一种鱼眼式的编辑策略。报纸都会报导本地新闻（比如持续进行的罢工）和其他地区的重要程度更高的新闻（比如贝鲁特的美国大使馆爆炸案）。

虽然这些实例的结果也可能有一些其他有趣的原因，我们得到如下结论：自然世界中的很多视角天生就具备鱼眼视图的特质。这说明了合适地应用鱼眼视图可以为浏览大量复杂结构的数据提供更好的视觉效果。

3 泛化鱼眼视图的一般形式

为了把鱼眼视图应用到交互界面的设计中，这个概念一定要更正式化和一般化。鱼眼视图只是这种更基本的用于大型结构视图策略的一个例子。这种基本的策略使用“兴趣指数”（DOI）来为结构中的每个节点赋值。兴趣指数是一个用来说明在某项任务中用户对于特定的某个节点的感兴趣程度的数值。对于一个可以容纳n个节点的视图，只需要选择兴趣指数最高的前n个节点来展示。

概括地说，只有发现合适的兴趣指标计算方法才能够产生成功的展示效果。对于兴趣指数，可能要把它划分的更细致才能了解其中奥义。泛化鱼眼视图中的兴趣指数分为两部分：其一是优先重要性，其二是距离。最简单的泛化鱼眼兴趣指数的公式如下：

DOIfisheye(x|.=y) = API(x) – D(x,y)

公式左边就是鱼眼视图模式下用户在x点的兴趣指数，y是当前用户的最直接关注点。API，也就是x点在全局结构中的优先重要性，D（x，y）是x点和y点之间的距离。这就是说兴趣指数是与优先重要性呈正相关关系，而与距离成负相关关系。（可以推测，兴趣指数的使用价值至少是基于距离与优先重要性的）

这个简单的公式使得鱼眼视图可以在任何有相应组成的结构中被定义。我们将以简单的树形结构为最直观的例子来阐述兴趣指数的公式，我们会发现这和之前提到的纽约人的视角十分不同。我们对于树形结构情有独钟，因为计算机内的很多大型结构都是以树来组织的：结构化的编程语言（比如LISP，PASCAL和C语言），分级组织的结构化文本（比如用户手册，法律条文），多种高度结构化的科技目录（比如生物学术语），分级的文件系统（比如UNIX），还有管理结构和分级的菜单系统等等。因此定义在树形结构上的兴趣指数可以便于我们在以上这些结构中应用鱼眼视图。

为了描述一棵树的组成，考虑x点到y点的距离D（x，y）为树中自然的距离dtree（x，y），也就是树中两个节点之间的路径。类似地，优先重要性可以用树中节点的深度来描述，可以是-dtree（x，root）—— x点距离根节点的距离，这是基于距离根节点越近重要程度越高的假设。（前面的负号直接给我们这样的感觉——离根节点越远越不重要）。这样一来，公式表达为：

DOIfisheye(tree)(x|.=y) = -(dtreee(x,y) + dtree(x,root))

图1 阐释了公式中的这两个部分，还有他们是如何运算得到这棵树中每个点的兴趣指数的。在结果中，数值越大说明对应的节点对于目光聚焦在y点的用户越感兴趣。因此，兴趣指数为-3的节点是当前这种情况下用户最感兴趣的，兴趣指数为-5的点则是用户其次感兴趣的，以此类推，随着兴趣指数减少，用户的感兴趣程度也不断下降。

所以通过选择一个阈值k，筛选兴趣指数大雨或等于k的节点来展示，就可以形成基于k而规模可变的鱼眼视图。比如，令k=-3，仅仅会选出最让用户感兴趣的节点，从图中可以看出，这些节点正是当前目光聚焦点的直系祖先，最终是根节点。（0级鱼眼视图）这个节点子集是最让用户感兴趣的是因为在这条直线上的节点的优先重要性逐个增长，正好弥补了他们在距离上的逐步疏远。如果阈值进一步减小以致于可以显示其次感兴趣的节点子集（１级鱼眼视图），包含了节点的祖先和亲兄弟。再扩大鱼眼视图的规模（阈值为－７，２级鱼眼视图），节点的表兄弟也被加了进来。这和鱼眼视图最本质的思想是一致的。选择阈值可以控制显示节点的感兴趣等级。

这些视图有一些有趣的特性。在一棵一般的树中，（１）鱼眼视图显示出对数级压缩的树；（２）由于兴趣指数集合有中凸和嵌套特性，因此计算时有一些快速的算法，时间复杂度和计算出的结果视图是一个级别的，而不是和整棵树一样；（３）视觉焦点从ｙ点转移到ｙ’时，视图的变化也是容易计算的，因为共同祖先的兴趣指数还是没有变化的；（４）用户移动视图所需的步骤也只是和插入的叶子节点的对数成正相关。这些性质提高了加强鱼眼视图的计算和交互效率的可能性。

４鱼眼视图交互界面

之前得到的针对树形结构的鱼眼视图兴趣指数计算公式被用来编写了一个展示结构化文本１级鱼眼视图的程序。这个例子展示了一个ｃ语言程序，见图3，图4，图5。（这是一个简单的用来计算逆波兰式整数加减运算的计算器程序）图3中的窗口视图展示出了很多细节，其中的一些不是非常有价值，比如标注出的那一行。（以>> 标出）上一个case的处理细节被强加到当前的页面中。而且没有什么导向性的信息。

另一方面，图4中的鱼眼视图展示了当前程序员正在关注一个for循环，这个循环发生在switch语句中的*e* case。这个switch语句中还包含其他四个case：+, -, q, default。Switch语句位于一个else代码块，else对应的是一个if语句。而这些又都在一个main函数的while循环内。可以这样认为，能够看到目前关注的焦点的同时能看到上下文性质的信息对于进行结构化语言编程的程序员是很有帮助的。图5比较了这两种视图的。这个方框表示的是图3显示的代码，加下划线的代码是图4显示的。虽然这两种视图都在中央位置显示局部细节的代码，但主要的区别是，图3中不是特别必需的一些细节代码被图4中的距离较远但是层次更高的突出上下文信息的代码取代。相关的程序视图最近也被一些语法驱动的程序编辑器中有所体现。