

基于 Haar 算法的云穿衣系统研究

王金环¹, 徐卫军², 李宝敏³

(1. 西安培华学院 智能科学与信息工程学院 计算机系, 陕西 西安 710125;

2. 西安工业大学图书馆, 陕西 西安 710021;

3. 西安工业大学 计算机学院, 陕西 西安 710021)

摘要:网购衣物已成为当前人们生活中一种非常普遍的消费方式。该项目借助计算机软件建立人体以及人脸的模型, 利用 Haar 算法对采集到的相关数据进行匹配, 匹配成功后, 通过硬件发射激光测量出深度场, 并结合光源方向、物体明暗情况及人体模型的匹配结果, 综合处理后, 确定人体的旋转角度; 然后将旋转角度赋予衣服模型, 将此渲染获得的图像与采集到的图像叠加, 通过显示屏, 得到了穿上衣服的效果。通过网格平均采样的方式, 获得图像中部分点的亮度, 根据亮度判断物体明暗情况, 并结合图像背景, 判断出光源位置。云服务器则负责接收商户提交的关于衣服的数据(照片、尺寸等), 建立衣服的 3D 模型, 并输出一些有关衣服与人体适配的参数, 提供给客户端进行下载。客户端需要结合参数合成衣服动画并渲染模型以及合成。

关键词: Haar 算法; 网格; 云服务器; 3D 模型; 渲染

中图分类号: TP274.3

文献标识码: A

文章编号: 1673-629X(2020)03-0214-07

doi: 10.3969/j.issn.1673-629X.2020.03.041

Research on Cloud Clothing System Based on Haar Algorithm

WANG Jin-huan¹, XU Wei-jun², LI Bao-min³

(1. Department of Computer, School of Intelligent Science and Information Engineering,

Xi'an Peihua University, Xi'an 710125, China;

2. Library of Xi'an Technological University, Xi'an 710021, China;

3. School of Computing, Xi'an Technological University, Xi'an 710021, China)

Abstract: Online shopping of clothes has become a popular way of consumption in people's lives. The human body and face models are built by computer software in this project, and the Haar algorithm is used to match relevant data collected. After successful matching, the depth field is measured by hardware emission laser. Combined with the direction of the light source, the light and shade of the object and the matching results of the human body model, the rotation angle of the human body is determined after comprehensive processing. Then, the rotation angle is given to the model of the clothes, and the image obtained from this rendering is superimposed with the image collected. Through the display screen, the effect of putting on the clothes is obtained. By means of grid average sampling, the brightness of the central part of the image can be obtained, and the brightness of the object can be judged according to the brightness, and the position of the light source can be determined according to the background of the image. The cloud server is responsible for receiving the data (photos, sizes, etc.) about the clothes submitted by the merchant, establishing the 3D model of the clothes, and outputting some parameters related to the fitting of clothes and human body for the client to download. The client needs to combine the parameters to synthesize the animation of the garment and render the model as well as the composition.

Key words: Haar algorithm; grid; cloud server; 3D model; rendering

0 引言

网购已成为当前人们生活中一种非常普遍的消费方式。然而在网购衣服的时候, 当你找到了自己心仪

的衣服款式, 而这种款式穿在自己身上的效果却不得知。以往的做法: 只能把衣服买回来试穿后才能感觉到是否美观、合身、称心。本项目解决了人们网购衣服

时的困扰,从而为人们网购带来了极大的便利。

从技术层面来讲,本项目利用计算机技术和一定的算法在云网的平台下对实物进行识别的尝试,对于许多通过网购而需要尝试的物品都可以通过该技术得到验证。这一技术改变了人们身临其境试穿的感官常规习惯和对网购物品体验的尝试,给人们以全新的理念和方式,给生活带来极大的方便。它是将智能识别技术^[1]在人们日常生活一些体验进行普及推广应用的一种尝试,其在社会和人们日常生活中具有很实用的意义和效应,尤其在当今信息主导社会发展,将带来一定程度的社会经济效益。

1 云穿衣系统总体设计

(1)云穿衣系统总体架构如图 1 所示。

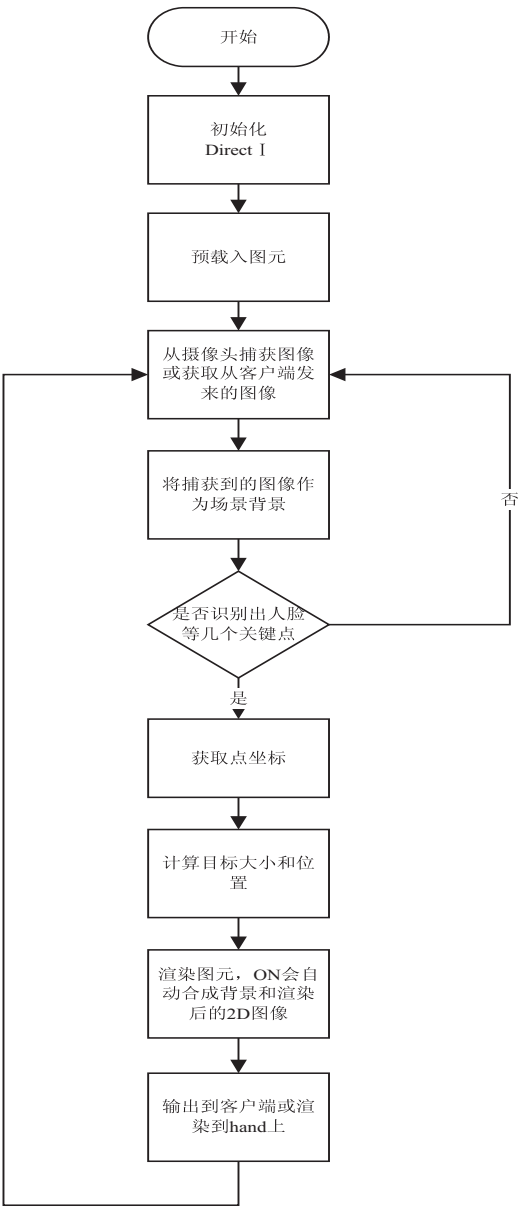


图 1 系统总体架构

●从摄像头获取图像;

- 从采集的图像中分析出人脸、人体的几个关键识别点^[2-3];
 - 将捕获的图像作为贴图,放置在场景的最后面一层;
 - 获取识别点的坐标^[4-5];
 - 计算目标大小和位置;
 - 根据目标大小和位置,针对 3D 图元进行变换;
 - 渲染输出图元;
 - 由 OM 自动合并图像。
- (2)实现部分详述。
- 预载入分类器特征文件;
 - 获取图像;
 - 利用积分图对每个像素计算特征值;
 - 通过决策树筛选并得到目标物体窗口;
 - 计算窗口大小及位置;
 - 根据窗口的大小、位置改变 3D 模型的坐标的缩放及转换;
 - 将获取的图像转换为 Texture2D 作为背景贴图;
 - 渲染 3D^[6-7]模型;
 - 由 DirectX 12^[8]的 Output Merger 自动合成;
 - 输出图像。

2 Haar 算法简介

2.1 Haar 算法

Haar 算法^[9]特征值反映了图像的灰度变化情况。Haar 特征有边缘特征、线性特征、中心特征和对角线特征。利用这些特征组合出仅有黑色矩形和白色矩形的特征模板,文中定义此模板的特征值为:白色矩形像素的和减去黑色矩形像素的和。以下进行举例:



图 2 Haar 算法特征值图像

对于图 2 中 A、B、D 三种模型,其特征值计算公式为 $v = \text{Sum 白} - \text{Sum 黑}$,而对于 C 模型,为了使两种矩形区域像素数目一致,需要为黑色像素和乘 2,因此公式为: $v = \text{Sum 白} - 2 * \text{Sum 黑}$ 。

计算过程中,通过调整模板的大小和位置,就可以得出很多特征。这些特征的值被称为“特征值”。由于特征量过于庞大,可以利用图 3 积分图来进行计算。

这里通过图像大小一样的一个二维数组来表示积分图,在该二维数组中, (x,y) 位置的值是在原始图像中从 $(0,0)$ 到 (x,y) 处的像素值的和。

对应图 3 中的 a、c 图所示的 haar 特征,计算公式如下:

$$\text{SAT}(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

$$\text{SAT}(x, y) = \text{SAT}(x, y-1) + \text{SAT}(x-1, y) + I(x, y) - \text{SAT}(x-1, y-1)$$

$$\text{RecSum}(r) = \text{SAT}(x-1, y-1) + \text{SAT}(x+w-1, y+h-1) - \text{SAT}(x-1, y+h-1) - \text{SAT}(x+w-1, y-1)$$

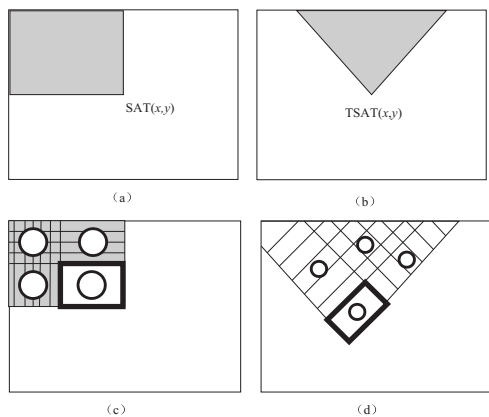


图 3 积分图

其中, SAT 表示积分图中的值, RecSum 表示 (x, y) 处的长为 w 、高为 h 的区域的和。以 RecSum 为基础来计算 haar 特征。同理, 对 b 图和 d 图对应的 haar 特征计算公式如下:

$$\text{RSAT}(x, y) = \sum_{y' \leq y, y' \leq y - |x-x'|} I(x', y')$$

$$\text{RSAT}(x, y) = \text{RSAT}(x-1, y-1) + \text{RSAT}(x+1, y-1) + I(x, y) - \text{RSAT}(x, y-2) + I(x, y) + I(x, y-1)$$

$$\text{RecSum}(r) = \text{RSAT}(x-h+w, y+w+h-1) + \text{RSAT}(x, y-1) - \text{RSAT}(x-h, y+h-1) - \text{RSAT}(x+w, y+w-1)$$

2.2 级联分类器 (CASCADE) 实现概述

2.2.1 弱分类器

弱分类器用来表示一个最基本的 Haar-like^[10] 特征, 通过对输入图像计算得到的 Haar-like 特征值与最初的弱分类器的特征值进行比较, 来判断所输入的图像是否为目标物体。弱分类器通过孵化训练就成为最优弱分类器。

$$h(x, f, p, \theta) = \begin{cases} 1, & pf(x) < p\theta \\ 0, & \text{其他} \end{cases}$$

一个弱分类器 h 是由 x 、 f 、 p 、 θ (x : 子窗口图像, f : 一个特征, θ : 阈值) 四部分组成。 p 的作用是控制不等式的方向, 使得不等式都是 $<$ 号, 形式方便。这里是通过分类和回归树进行识别, 如图 4 所示。

在分类的应用中, 每一个非叶子节点都表示一种判断, 每一条路径都代表一种判断的输出, 每一个叶子节点代表一种类别, 并作为最终判断的结果。

一个弱分类器就是图 4 类似的决策树, 最基本的弱分类器只包含一个 Haar-like 特征, 也就是说它的决策树只有一层, 被称为树桩 (stump)。

要注意的是如何来确定每一个节点判断的输出, 在对输入图像的特征值与弱分类器中的特征进行比较时, 需要有一个阈值才可以。判定其为人脸的条件是: 只有当输入图像的特征值大于该阈值时才成立。对最优弱分类器的训练过程就是在寻找一个合适的分类器的阈值, 使得该分类器对所有现有的样本的判读误差降为最低。获得阈值的操作过程如下:

(1) 对于每个特征 f , 计算所有训练样本的特征值, 并将其排序。

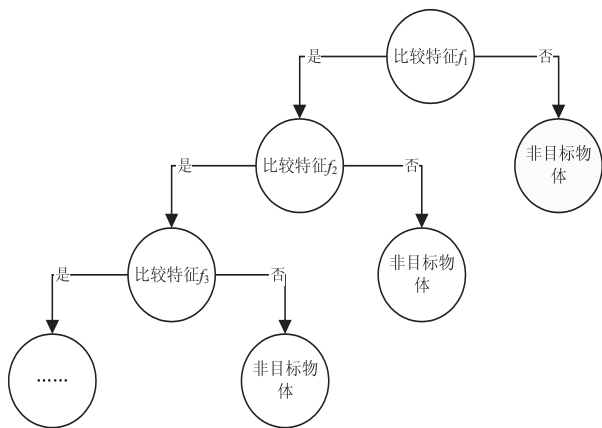


图 4 回归树

(2) 对排好序的特征值扫描一遍, 对每个元素, 计算求得以下四个值:

- (a) 全部人脸样本的权重的和 t_1 ;
- (b) 全部非人脸样本的权重的和 t_0 ;
- (c) 在此元素之前的人脸样本的权重的和 s_1 ;
- (d) 在此元素之前的非人脸样本的权重的和 s_0 。

(3) 最终在获得每个元素的分类误差对应表中寻找 r 值最小的元素, 并将该元素作为最优阈值。

$$r = \min(s_1 + (t_0 - s_0), (s_0 + (t_1 - s_1)))$$

2.2.2 强分类器

强分类器需要 T 轮的迭代, 具体操作如下:

(1) 给定训练样本集 S (共 N 个样本), 其中 X 和 Y 分别对应于正样本和负样本, T 为训练的最大循环次数;

(2) 初始化样本权重为 $1/N$, 即为训练样本的初始概率分布;

(3) 第一次迭代训练 N 个样本, 得到第一个最优弱分类器, 步骤见 2.2.1 节

(4) 提高上一轮中被误判的样本的权重;

(5) 将新的样本和上次被误判的样本放在一起进行新一轮的训练。

(6) 循环执行步骤 4-步骤 5, T 轮后得到 T 个最优

弱分类器。

(7)对 T 个最优弱分类器组合得到强分类器。

组合方式如下:

$$C(x)=\begin{cases}1 & \sum_{i=1}^T a_i h_i(x) \geq \frac{1}{2} \sum_{i=1}^T a_i \\ 0 & \end{cases}$$

$$a_i = \log \frac{1}{\beta_i}$$

强分类器的结果就是由所有弱分类器加权求和的结果与其分类结果平均值进行比较得到最终结果。

2.2.3 目标 Haar 分类器

为了使得检测的结果更加准确,需要进行级联分类器训练。这涉及到 Haar 分类器的另一个体系—检测体系。检测体系是以现实中的一幅大图像作为输入,然后对图像进行多区域、多尺度的检测。由于训练的时候用的照片一般都是 $20 * 20\text{ mm}$ 左右的小图片,所以对于大的目标物体,还需要进行多尺度的检测。在区域放大的过程中会出现同一个人脸被多次检测,这还需要进行区域的合并。

无论哪一种搜索方法,都会为输入的图像输出大量的子窗口图像,这些子窗口图像,经过筛选级联分类器多次地被每一个节点筛选、抛弃或通过。它的结构如图 5 所示。

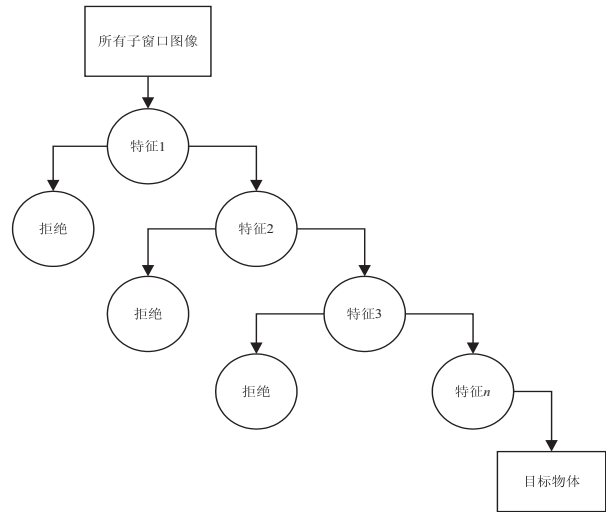


图 5 Haar 分类器结构

级联强分类器的策略一般是:将若干个强分类器由简单到复杂排列,经过训练使得每个强分类器获得较高检测率和低的误识率,比如几乎 99% 的目标物体可以通过,但 50% 的非目标物体也可以通过,这样如果有 20 个强分类器级联,那么它们的总识别率为 0.99^{*20} ,约为 98%,错误接受率也仅为 0.5^{*20} ,约为 0.000 1%。这样的效果就可以满足现实的需要。

设 K 是一个级联检测器的层数, D 是该级联分类器的检测率, F 是该级联分类器的误识率, d_i 是第 i 层强分类器的检测率, f_i 是第 i 层强分类器的误识率。如果要训练一个级联分类器,达到给定的 F 值和 D

值,只需要训练出每层的 d 值和 f 值:

$$d^k = D, f^k = F$$

级联分类器的要点就是如何训练每层强分类器的 d 值和 f 值使其达到指定要求。级联分类器在训练时要考虑弱分类器的个数和计算时间的平衡,以及强分类器检测率和误识率之间的平衡。

3 云穿衣系统 3D 模型

3D 模型:通过调用 DirectX3D^[11] 接口实现对 3D 模型的渲染。

3.1 图形管线综述

图形管线是运行在图形硬件上处理流程中的一部分,叫做 Stages。将数据推入图形管线就可以获取 3D 场景的 2D 图像。还可以利用 Stream Output Stage 来流式输出处理后的集合体。某些图形管线可以被编程,可以被编程的图形管线被称作 Shaders,编程图形管线使用高级着色语言(HLSL)。

3.1.1 图形管线中的着色器

图形管线中的着色器如图 6 所示。

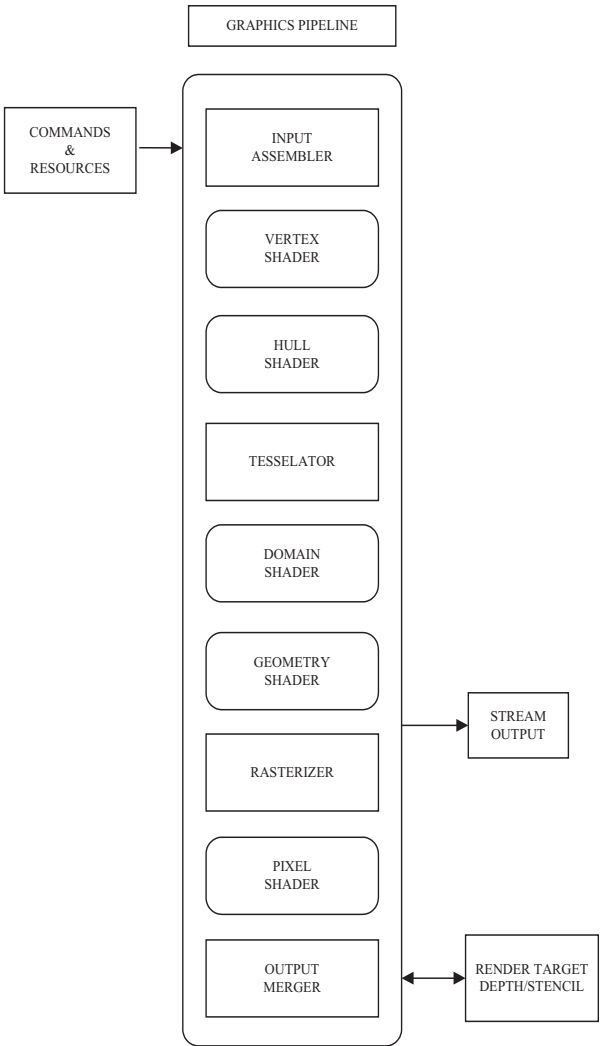


图 6 图形管线着色图

3.1.2 着色器各部分的功能

(1) The Compute Shader (CS)。

CS 专门用来进行任何类型的计算,CS 利用 CPU 平行计算的优势,可以处理很多对性能要求较高的计算,如碰撞检测等。

(2) Input Assembler (IA) Stage。

这是图形管线中的第一个 Stage,这个 Stage 是静态的,不可编程的。用 D3D Device 配置 IA,以便 IA 可以理解如何创建图元。需要提供输入布局给 IA,以便 IA 了解如何读取顶点数据。IA 还有一个功能,就是将图元放到一起,并附加到系统根据图元生成的数据上。这些数据被称为“语义”。例如输入布局可以是这样的:

```
D 3D11_INPUT_ELEMENT_DESC layout[ ] =
{
    { "POSITION", 0, DXGI_FORMAT_R32G32B32_FLOAT,
    0, 0,
    D 3D11_INPUT_PER_VERTEX_DATA, 0},
};
```

这个输入布局告诉 IA,每个顶点缓冲区有一个元素,语义为“Position”。它还说明了这个元素从第一个顶点字节包括了 3 个浮点数,每一个是 32 位,4 个字节。

(3) Vertex Shader (VS) Stage。

VS 是第一个可编程 Stage (Shader),必须为这个 Stage 编写独立的程序。VS Stage 用来处理所有图元的顶点数据。VS 可以处理顶点的变换、缩放、光照、置换贴图 etc。

最简单的 VS 程序如下所示,它直接将输入的顶点输出。

```
float4 main( float4 pos:POSITION ) : SV_POSITION
{
    return pos;
}
```

Vertex shader 直接返回了输入的位置。注意参数中的 POSITION 在 Pos 的后面,这是语义的简单的例子。在输入给 IA 的顶点缓冲区布局中说明了数据的语义为 POSITION,于是在这里通过 POSITION 获取。

(4) Hull Shader (HS) Stage。

这是三个可选着色器中的第一个着色器。三个可选着色器统称为 Tessellation Stages,包含了 Hull Stage、Tessellator、Domain Shader,它们一起工作并实现了如何进行曲面细分。

曲面细分取出一个图元对象,然后划分成多个小部分,以便增加模型的细节。它可以以极快的速度在图元出现在屏幕之前在 GPU 上创建新的图元。

Hull Shader 是一个可编程的 Stage,Hull Shader 可

以为曲面细分后的图元添加细节,它会将数据发送给 Tessellator Stage 和 Domain Shader Stage。

(5) Tessellator (TS) Stage。

Tessellator Stage 是曲面细分的第二 Tessllator Stage 不可编程。它从 Hull Shader 中获取数据,然后将图元进行细分,最后将数据输出给 Domain Shader。

(6) Domain Shader (DS) Stage。

这是曲面细分的第三步。这是一个可编程的 Stage。这个 Stage 从 Hull Shader 中取出顶点位置,然后变换从 Tessllator Stage 中取出顶点来为图元添加细节。

(7) Geometry Shader (GS) Stage。

这是另一个可选的可编程 Shader。它接受图元作为输入,然后输出另一个图元,与 Vertex Shader 不同的地方在于,这个 Shader 可以利用一个图元创建另一个图元,而 Vertex Shader 不能创建新的图元。这个 Shader 常常用来创建粒子特效。

(8) Stream Output (SO) Stage。

这个 Stage 用来从管线中获得顶点数据。

(9) Rasterizer Stage (RS)。

光栅器输入向量信息(图形或图元),然后将它们转换为像素。它还将不出现在视口中的图元裁剪掉。

(10) Pixel Shader (PS) Stage。

这个 Stage 计算和修改每个将会在屏幕上出现的像素,比如光照信息等,这是一个可编程 Shader。

PS 的功能在于,计算每个 Pixel Fragment 的最终颜色,Pixel Fragment 是每个可能被显示在屏幕上的像素。举个例子,有一个单面的矩形在一个单面的圆后面,矩形的像素和圆的像素都是 Pixel Fragments,它们都有机会被写入屏幕,它发现圆的深度数据小于矩形的,于是只有圆的像素会被显示到屏幕上。PS 将会输出一个 4D 的颜色数据。举一个简单的例子:

```
float4 main( ) :SV_TARGET
{
    return float4( 1.0f,1.0f,1.0f,1.0f );
}
```

这个像素着色器直接将每个将会出现在屏幕上的像素设置为白色。

(11) Output Merger (OM) Stage。

这是渲染管线的最后一个部分。这个 Stage 将会获取每个 Pixel Fragment 的深度、镂空信息,来决定最终哪些像素可以被渲染到渲染目标上。

3.2 Direct X12 工作

DirectX12 工作主要包括以下四个方面:

(1) Pipeline State Objects (PSO)。

PSO 指的是 ID3D12PipelineState 这个接口,利用

Device 的 CreateGraphics Pipeline State() 方法来创建。这个结构体将会决定渲染管线的状态。

(2) The Device。

Device 指的是 ID3D12Device 接口。Device 是一个用来创建 CL, PSO, Root Signatures, CA, Command Queues, Fences, Resources, Descriptors 以及 Descriptor Heaps 的虚拟适配器。通过 D3D12CreateDevice() 函数来创建 Device。

(3) Command Lists (CL)。

CL 指的是 ID3D12CommandList 接口,使用 CL 来分配需要在 GPU 上执行的命令。

(4) Bundles。

Bundles 指的是 ID3D12CommandList 接口, Bundles 是一组经常被重用的命令,因为 CPU 在创建命令的时候很浪费时间,因此命令的重用对提升效率很有作用。

3.3 Command Queues (CQ)

CQ 指的是 ID3D12CommandQueue 接口,CQ 还用来更新资源映射。

3.3.1 Command Allocators (CA)

CA 指的是 D3D12CommandAllocator 接口,是显存中 CL 和 Bundles 所储存的空间。

3.3.2 Resource

Resource 包括在构造场景时需要用到的数据。它是储存了几何体、贴图、着色器数据的可以被图形管线访问的大块内存。

3.4 资源类型

资源类型是指资源所保存的数据的类型。包括: Texture1D、Texture1DArray、Texture2D、Texture2DArray、Texture2DMS、Texture2DMSArray、Texture3D、Buffers (ID3D12Resource)。

3.5 Descriptors (Resource Views)

Descriptors 是一个结构体,用来告诉 Shaders 哪里可以找到资源,以及如何解释资源数据。可以为同一个资源创建多个描述符,用来以不同的方式传递给不同的 Stage。

举个例子,可以创建一个 Texture2D 资源,然后创建一个渲染目标视图以便使用资源作为输出缓存的管线。描述符只能被放置在描述符堆中,不能储存在内存中。

3.5.1 Descriptor Tables (DT)

Descriptor Tables 是一个描述符数组。Shader 可以从描述符堆中访问描述符,通过 Root Signature 的 Descriptor Tables 的索引。为了从 Shader 中访问描述符,应该给 Root Signatures 的 Descriptro Tables 进行索引。

3.5.2 Descriptor Heaps (DH)

描述符堆表示的是 ID3D12 Descriptor Heap 接口,是一个占用大段内存的描述符列表。

3.6 Root Signatures (RS)

Root Signatures 定义了 Shader 访问的数据(资源)。Root Signatures 类似于一个函数的参数列表,函数就是 Shader,参数列表就是 Shader 访问的数据的类型。

(1) Root Constants 是一个内联的 32 位数据 (DWORD)。这些数据直接储存在 Root Signature 中,因为内存限制了 Root Signature,所以把 Shader 可以访问的经常改变的数值放置到这里。

(2) Root Descriptors 是可以被 Shader 经常访问的内联的 Descriptors。占用 64 位虚拟地址 (2 DWORDS)。

(3) Descriptor Tables 由偏移量和长度组成,位于 Descriptor Heap 中,Descriptor Tables 只有 32 位。不限制其中保存的 Descriptor 数量。

3.7 Resource Barriers

Resource Barriers 被用来改变资源的状态或者资源的子资源的使用。

有三种资源的种类:

(1) Transition Barrier 用来转换资源以及子资源的状态。

(2) Aliasing Barriers 与 Tiled Resources 一起使用。

(3) UAV Barriers 用来确保读写操作已经完成。

3.8 Fences 与 Fence Events

Fences 与 Fence Events 会指示 GPU 是否在执行 CQ。Fences 指的是 ID3D12Fence 接口,通过 Device 的 CreateFence() 来创建,Fence Events 通过 CreateEvent() 方法来创建。

4 关键技术实现

通过计算机软件建立人体以及人脸的模型,利用 Haar 算法对采集到的数据进行匹配,匹配成功后则判定该物体为人体,通过硬件发射激光测量出深度场,并结合光源方向、物体明暗情况、人体模型的匹配结果,综合判定人体的旋转角度,然后将旋转角度赋予衣服模型,将渲染获得的图像与采集到的图像^[12]叠加,最后显示到显示屏上,就得到了穿上衣服的效果。

通过网格平均采样的方式,获得图像中部分点的亮度,根据亮度判断物体明暗情况,并结合图像背景,判断出光源位置。

云服务器负责接收商户提交的关于衣服的数据(照片、尺寸等),建立出衣服的 3D 模型,并输出一些有关衣服与人体适配的参数,提供给客户端进行下载,

客户端需要结合参数合成衣服的动画并渲染模型以及合成。

具体要解决的技术问题:(1)首先解决采集物的信息,采集手段与信息处理;(2)对采集到的人与物等信息进行识别;(3)在云平台上^[13]进行人与衣服匹配的信息处理;(4)将处理后的信息提交给人去鉴赏。

5 实验效果

从摄像头或客户端捕获图像,见图7。

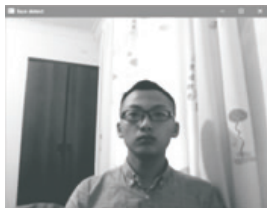


图7 捕获图像

从收到的图像中识别出人脸、肩膀部位,见图8。



图8 识别图像

基于机器学习^[14]以及 Haar 算法识别出特定的物体,如人脸,调用 DirectX 进行渲染,通过 DirectX 12 用显卡,进行图元的渲染。

由 DirectX 12 渲染管线中的 OM 直接合成图像,然后绘制到 hwnd 上或输出图片,见图9。



图9 合并图像

6 结束语

本项目在实践初期经常遇到识别不准的问题。经过研究发现,是由于样本数量不足导致的,后期将精力放在了样本的采集上,使得识别精确度大幅提高。

(1)利用物体识别算法对物体大小的侦测并不完

全靠谱,应当对物体的某些小区域进行识别,根据多个小区域之间的距离计算出物体的大小,这才是合适的做法。

(2)在物体识别的过程中,摄像头扮演的角色十分重要,因此,需要高精度摄像头,摄像头产生的杂色会严重影响到物体识别的精确度。

(3)这是智能识别技术在网购衣物方面的一次尝试,尚存在许多不尽人意的地方,有待今后更进一步的研究和学习,使其更加趋于完善和成熟。

参考文献:

- [1] 杨欣.自动人脸识别若干关键问题研究[D].南京:东南大学,2007.
- [2] 张翠平,苏光大.人脸识别技术综述[J].中国图象图形学报,2000,5(11):885-894.
- [3] 姜睿.基于知识的人脸检测与人脸识别系统研究[D].西安:西北工业大学,2004.
- [4] 段锦.人脸自动识别中若干问题的研究[D].长春:吉林大学,2004.
- [5] 马颖哲.多视角人脸检测技术的研究[D].阜新:辽宁工程技术大学,2010.
- [6] 龙开文.基于模板匹配的人脸检测[D].成都:四川大学,2005.
- [7] 张晶晶.人脸检测与人脸特征定位技术研究[D].南宁:广西民族大学,2010.
- [8] LUNA F. Introduction to 3D game programming with DirectX 12[M]. Plano, TX, USA: Wordware Publishing Inc., 2015.
- [9] 魏京天,王军,王玉楠,等.基于 Haar-NMF 特征和级联 AdaBoost 的脱岗检测算法[J].测控技术,2019,38(2):50-55.
- [10] 王庆伟,应自炉.一种基于 Haar-Like T 特征的人脸检测算法[J].模糊识别和人工智能,2015,28(1):35-41.
- [11] ZINK J, PETTINEO M, HOXLEY J. Practical rendering & computation with direct 3D 11[M]. [s. l.]: A K Peters/CRC Press, 2011.
- [12] 周明全,耿国华,韦娜.基于内容图像检索[M].北京:清华大学出版社,2007.
- [13] 张强.云计算时代的平台战略[J].电脑知识与技术,2010,6(6):1350-1352.
- [14] BISHOP C. Pattern recognition and machine learning[M]. Berlin: Springer, 2007.