实时渲染超分插帧技术综述

冯泽森，张潍韬，陈彦齐，张嘉伟，郭延文，过洁*

(计算机软件新技术全国重点实验室(南京大学)　南京　210033)
(guojie@nju.edu.cn)

摘　要：随着实时渲染场景复杂度的不断提升与显示规格的持续升级，图形硬件性能日益逼近瓶颈。超分辨率与插帧技术能够在低分辨率、低帧率渲染条件下，实现接近原生渲染的画质效果。与通用场景中的超分与插帧不同，实时渲染环境下的该类技术面临更为严苛的实时性要求，同时可充分利用渲染管线提供的丰富辅助信息（如深度、运动向量等），在虚拟现实、电子游戏等领域发挥着重要作用。文中对实时渲染中的超分与插帧技术进行系统综述。首先阐述超分与插帧的基础定义与基本思路；随后将现有方法划分为仅超分、仅插帧以及超分插帧联合 3 类，分别进行介绍与对比分析；最后总结当前技术发展现状，并探讨未来可能的研究方向，包括基于场景理解的超分与插帧、算法硬件化、以及超分插帧与正常渲染的并行处理等。

关键词：实时渲染；超分辨率；帧生成；插帧
中图法分类号：TP391.41　　**DOI：**10.3724/SP.J.1089.2025-00502

Overview of Real-Time Rendering Super-Resolution and Frame Generation

Feng Zesen, Zhang Weitao, Chen Yanqi, Zhang Jiawei, Guo Yanwen, and Guo Jie*

(*State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing*　210033)

Abstract: As real-time rendering incorporates increasingly complex scene content and higher display specifications, graphics devices are approaching performance bottlenecks. Super-resolution (SR) and frame generation (FG) solutions can achieve visual effect comparable to native rendering while operating at low resolutions and frame rates. Different from general SR and FG methods, SR and FG for real-time rendering face stricter performance requirements while accessing more comprehensive auxiliary information from rendering engines (e.g., depth, motion vectors), thus playing a crucial role in fields such as virtual reality (VR) and video games. This paper provides a systematic review of SR and FG methods for real-time rendering. First, the basic definitions and core concepts of SR and FG are introduced. Subsequently, existing methods are classified into three categories — SR-only, FG-only, and joint SR-FG — then analyzed and compared. Finally, the research progress of SR and FG methods for real-time rendering is summarized, and potential future research directions are discussed, including scene-aware SR and FG, hardware implementation of relevant algorithms, and parallelization of SR/FG with normal rendering.

Key words: real-time rendering; super-resolution; frame generation; frame interpolation

收稿日期：2025-12-04；修回日期：2026-03-23。**基金项目：**国家自然科学基金（61972194，62032011）；教育部基础学科和交叉学科突破计划（JYB2025XDXM118）；江苏省自然科学基金（BK20211147）。**冯泽森**（2000—），男，硕士研究生，主要研究方向为计算机图形学；**张潍韬**（2003—），男，硕士研究生，主要研究方向为计算机图形学；**陈彦齐**（2005—），男，在校学生，主要研究方向为计算机图形学；**张嘉伟**（2002—），男，硕士研究生，主要研究方向为计算机图形学；**郭延文**（1980—），男，博士，教授，博士生导师，CCF 会员，主要研究方向为计算机图形学；**过洁**（1986—），男，博士，长聘副教授，博士生导师，CCF 高级会员，论文通信作者，主要研究方向为计算机图形学、虚拟现实。

当前，计算机图形学领域向图形应用中引入了多种高质量和真实感效果，并提出大量优化方法，以便将这些效果在实时的帧率（至少 30 帧/s）下呈现。近年来，随着用户对高质量视觉效果、低延迟交互的需求不断增长，以及高分辨率、高刷新率的显示设备成为市场主流，内容质量与显示质量的要求变得更加严格。对于图形应用而言，意味着每秒需要在更多的像素上进行更加复杂的计算，而设备的处理性能逐渐无法支持这样规模的计算任务。对于供电有限的平板、手机等移动设备，功耗和续航的限制进一步压缩了高质量图形应用的表现。在此背景下，研究人员提出了一系列面向实时渲染的超分与插帧方法，首先在较低分辨率和帧率下进行原生绘制，然后通过多种途径将绘制内容提升至目标分辨率与帧率。与自然照片的超分或视频流插帧不同，实时渲染的超分任务与插帧方法有更加严格的时间限制和更多可用的信息。由于超分与插帧一般在实时渲染的后处理阶段进行，考虑到其他模拟和渲染的耗时，因此一次超分或插帧计算需要在少于 1 帧的时间内完成（30 帧/s 下 33 ms 以内）。在帧序列之外，实时渲染应用可以向超分插帧算法提供更丰富且准确的信息，如相机参数、物体的运动状态和场景深度，甚至是渲染过程的中间数据，这些都有助于提高超分与插帧的质量。在评价超分或插帧算法的质量时，除了每帧内容的正确还原之外，帧序列在时间上的稳定性也是一个重要参考，否则会出现画面闪烁、抖动，或者细小物体丢失等问题。

目前，多数超分与插帧算法同时使用空域与时域信息进行辅助。空域上，算法选择当前帧局部或全局的像素样本；时域上，算法从历史帧中收集可用的像素。比较有代表性的方法有 AMD 公司推出的 FSR 技术[1]，NVIDIA 公司推出的 DLSS 技术[2]，它们都经历了从空域超分走向时域-空域联合的过程。当前，实时渲染超分插帧的研究热点集中在更准确且有效地利用渲染帧的时序相关性、平衡人工智能（artificial intelligence，AI）辅助，与实时性能、算法在移动端上的高效运行等方面。

超分与插帧使用更少的像素信息合成目标帧序列，本质上是完成在时域和空域上进行重建的任务。本文首先给出超分与插帧任务的形式化定义，介绍实时渲染中超分与插帧问题的特殊性和解决？思路；然后对输入数据的依赖进行分类，介绍应用于实时渲染的仅超分、仅插帧和超分插帧联合方法；最后总结分析现有超分、插帧与超分插帧联合 3 类方法的优缺点，指出未来可能的研究方向。

1 背景知识

1.1 超分与插帧的定义

在实时渲染中，一个渲染帧通常从分辨率、时刻和内容 3 方面描述。分辨率指一帧的像素数量，以宽乘以高表示，本文中将渲染帧分辨率分为低清（low resolution，LR）和高清（high resolution，HR）。因为涉及场景模拟以及 CPU 与 GPU 之间的交互，所以时间因素在实时渲染中比较复杂。如果没有特别说明，本文中一帧的“时刻”指 GPU 开始渲染的时间点，帧率则为单位时间内可以呈现在屏幕上的帧数量；一帧的内容指屏幕空间（图像空间）内像素的含义，除了场景颜色 I 外，帧内容还包括场景深度 D 和运动向量（motion vector，MV），在延迟渲染管线中还包括空间位置 P、物体表面法向量 $\boldsymbol{N}$、材质粗糙度 R 等 G-buffer 内容。综上所述，一个 t 时刻的低清场景颜色帧可以被描述为 $f_t^{I,\mathrm{LR}}$。

超分与插帧则分别是在（屏幕空间）分辨率和时间上对渲染内容进行超采样的任务，可以表示为在一系列渲染帧上进行空域和时域采样的变换

$$f_t^{I,\mathrm{HR}} = F\left(f_{\cdots}^{\cdot,\mathrm{LR}}, f_{\cdots}^{\cdot,\mathrm{HR}}, S, T\right)。$$

其中，$f_t^{I,\mathrm{HR}}$ 表示目标帧；F 表示超分插帧变换，其输入包含低清渲染帧序列 $f_{\cdots}^{\cdot,\mathrm{LR}}$、高清渲染帧序列 $f_{\cdots}^{\cdot,\mathrm{HR}}$、空域采样操作 S 和时域采样操作 T 这 4 个部分。

1.2 渲染内容辅助超分超帧

渲染帧序列来自基于几何数据的着色计算，与自然的视频帧序列相比，渲染帧包含许多可以用于辅助采样任务的数据，如使用准确的场景深度辅助识别几何信息，或者通过 MV 复用相邻帧的数据。在延迟渲染管线[3]中，也可以通过 G-buffer 获取像素的空间位置、法向量等更加精确的几何信息。充分利用这些数据可以极大地提升超分插帧的质量[4]，但也需要引擎或用户进行相应程度的管线修改和数据开放，同时增加算法的吞吐量和计算量，这些都会影响实时渲染的性能。因此，无 G-buffer 辅助的超分与插帧任务也是一个研究方向[5-7]。

除了渲染内容上的辅助，目标空间上的辅助也可以提高超分与插帧的效果，例如，许多超分方法使用高清的 G-buffer 作为辅助[8-9]，多数插帧方法也从目标帧的 G-buffer 中提取信息[10-12]，或者使用目标时刻的 MV 辅助运动估计[11]。

1.3 内插与外插

现有的插帧（帧生成）方法一般分为内插

（Interpolation）与外插（Extrapolation）2 类，主要的区别在于输入帧序列的时刻。以输出时刻 t 的目标帧颜色 $f_t^{I,\cdot}$ 为例，内插的输入帧序列指前后相邻的渲染帧，即 $f_t^{\cdot,\cdot}$ 和 $f_{t+1}^{\cdot,\cdot}$；而外插的输入序列为前 k 个渲染帧，即 $f_{[t-k,\cdots,t]}^{\cdot,\cdot}$。

有了“未来一帧”提供的信息，内插方法可以达到比较高的质量，但由于需要等待时序上的下一帧渲染完成，因此每个渲染帧都需要一定的延迟才会显示到屏幕上。为了降低这个延迟，商用的内插方法通常在硬件驱动层面做大量优化，或者配合独立的交换链来调整每帧呈现到屏幕上的时机。外插方法在质量上普遍不如内插，但其不会引入额外的时延，更加适用于对时延敏感的场景，也便于集成或在平台之间迁移。在同样的渲染、插帧开销和显示帧率下，外插与内插引入的时延对比如图 1 所示。可以看出，内插引入的时延较高。

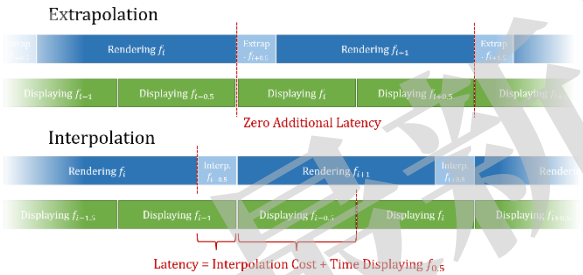
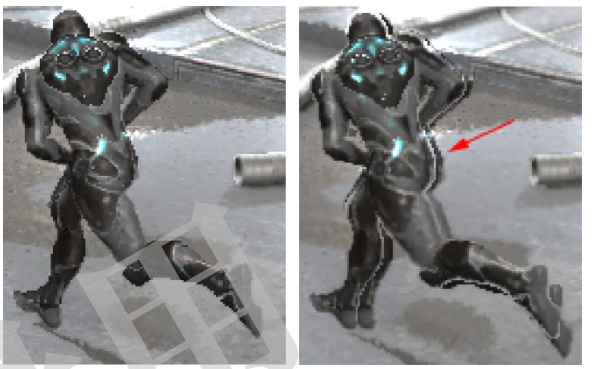


图 1　外插与内插引入的时延对比[10]

1.4　MV 与时序累积样本

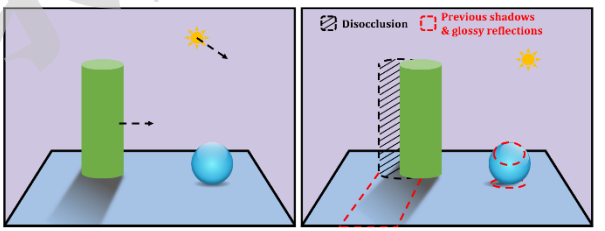
在多个时刻的渲染帧上进行采样时，需要将渲染帧的内容通过重投影操作对齐到同一个时刻的屏幕空间下。对于时刻 t，$\mathbf{MV}_{t-1\to t}$ 表示场景内的几何物体在屏幕空间内从 $t-1$ 时刻到 t 时刻的相对位移，由引擎根据相机和场景内物体的运动计算得到，因此可以代表准确的几何运动，这是它与光流最大的区别。通过 MV 将颜色帧从 $t-1$ 时刻重投影到 t 时刻可以表示为 $\mathbf{MV}_{t-1\to t}\times f_{t-1}^{I}$。重投影可以由历史帧到当前帧或者由当前帧到历史帧，也可以迭代多次。

然而，使用 MV 指导采样存在 2 个问题：（1）可见性变化导致历史样本的有效性变化。MV 是物体的位移在屏幕空间内的投影，如果上一帧被遮挡的物体在当前中暴露出来，它会拥有一个 MV，但是根据向量无法在上一帧中找到其对应的像素，这种情况一般在运动的前景物体经过静止或较慢的背景时出现。此时，直接根据 MV 找到的是前景在上一帧的像素，则采样的结果是运动物体背后出现它的鬼影，如图 2 所示。通过比较深度或 G-buffer 内容等方法可以识别出这些区域[9-10,13]，但是由于缺少一个维度，这些方法都不是完全准确的。（2）MV 只表达了几何的运动关系，而一般的渲染内容除了几何效果，还包含阴影、高光等着色效果以及粒子效果。如图 3 所示，这些效果往往与几何运动不一致，无法由 MV 处理。



a. 实际渲染结果　　b. 普通重投影结果

图 2　普通重投影的鬼影现象[10]



f_i　　$f_{i+0.5}$

图 3　重投影无法处理可见性变化和着色变化[10]

为了解决上述问题，一些方法仍然基于 MV，通过做额外的修正表达非几何效果的运动或者缓解鬼影[10]；也有的方法使用更复杂的采样策略处理鬼影问题[5,11]；还有一些方法引入光流或者表达类似关系的映射辅助采样[1-2,12]。

在使用 MV 的基础上，还可以通过 jitter 丰富时序样本[7,14]，即通过对相机附加微小的抖动使每帧采样的像素内容都有些许不同。

2　实时渲染超分方法

实时渲染超分与普通图像超分在时序累计和 G-buffer 辅助方面是不同的。时序累积指方法可以通过 MV 将历史帧信息重投影到当前帧，为超分算法提供更多像素样本；G-buffer 辅助则利用 G-buffer 提供的准确的几何数据提高算法对时域上的几何变化与空域上的细小结构、高光、阴影等内容的处理能力。

2010 年，Herzog 等[15]联合渲染内容的空域和时域信息对低清渲染帧进行超分辨率，针对在连续的渲染帧序列中静态元素的质量最关键，而动态元素可以允许部分细节损失的特性，提出一种基于权重的超分算法 S-T Upsampling，将时域上采样与空域上采样的权重计算相结合，高清当前帧的像素由低清当前帧和历史帧根据空域几何感知权重、历史帧权重和时序衰减权重共同决定；对于静态场景，选择复用时序样本，当发生剧烈运动时空间一致性样本的权重增加。虽然在运动场景下会损失部分视觉精度，但是该算法能保持超分结果的时序稳定；除了常用的颜色、深度、法向等渲染数据，还利用材质 ID 识别时序上新暴露出来的区域，提高重投影的样本质量。

在 S-T Upsampling 算法的基础上，已有的实时渲染超分方法在时域和空域的利用上进行了深入的探索。本文将以是否依赖 G-buffer 这一关键数据进行介绍，并在最后集中介绍工业界常用的实时渲染超分方法。实时渲染超分方法对高清数据的依赖情况、时空信息的利用和优势场景如表 1 所示。

表 1　实时渲染超分方法对高清数据的依赖情况、对时空信息的利用和优势场景

方法	依赖的高清数据	时域信息利用	空域信息利用	优势场景
S-T Upsampling[15]	G-buffer	历史帧按权重混合	邻域像素按权重混合	基于规则，计算开销低
NSRR[16]		历史帧按权重混合	零上采样保留高频信息	不需要 jitter 适配，集成难度低
Classifier Guided[17]		超分结果复用，鬼影掩模	走样掩模	时序稳定性高
FuseSR[8]	G-buffer	超分结果复用	高清 G-buffer 辅助几何重建，对齐到低清空间计算	几何结构超分质量高
Subpixel Sampling Reconstruction[18]	G-buffer	超分结果与特征复用，时序分摊采样	掩模标记有效子像素样本	适用于光线追踪降噪
Efficient NSR[19]		超分结果与特征复用，jitter 指导的时序信息累积	调整 mipmap 偏移量增加低清帧的高频信息	计算开销较低
ArbSR[9]	G-buffer	超分结果复用，时域掩模标记变化区域	高清 G-buffer 提取高低频信息，空域掩模标记高频区域	适用于高倍率超分
多尺度特征融合[20]	G-buffer		各编码层融合低清颜色帧和高清 G-buffer 特征	无伪影瑕疵，几何结构超分质量高
MNSS[7]		超分结果复用，特殊设计的 jitter 模式	抗锯齿结果插入高清历史帧	计算开销低，适用于移动端
NFSR[21]		超分结果复用，jitter 指导的时序信息累积	根据视点选择超分质量	适用于 VR 相关超分

2.1　依赖 G-buffer 的实时渲染超分方法

利用 G-buffer 的最直观的方法是挖掘其空域信息。Zhong 等[8]提出的 FuseSR 从高清空间的 G-buffer 中提取高频细节，辅助网络进行精细结构的重建，其整体流程如图 4 所示。FuseSR 使用了一个高效的 H-Net 结构，通过 S2D 和 D2S 操作将高清 G-buffer 数据对齐到低清空间处理；在 G-buffer 内容的利用上，FuseSR 利用 BRDF 解调制使网络集中于对“能量传输”的预测，提高对光照效果的处理能力。

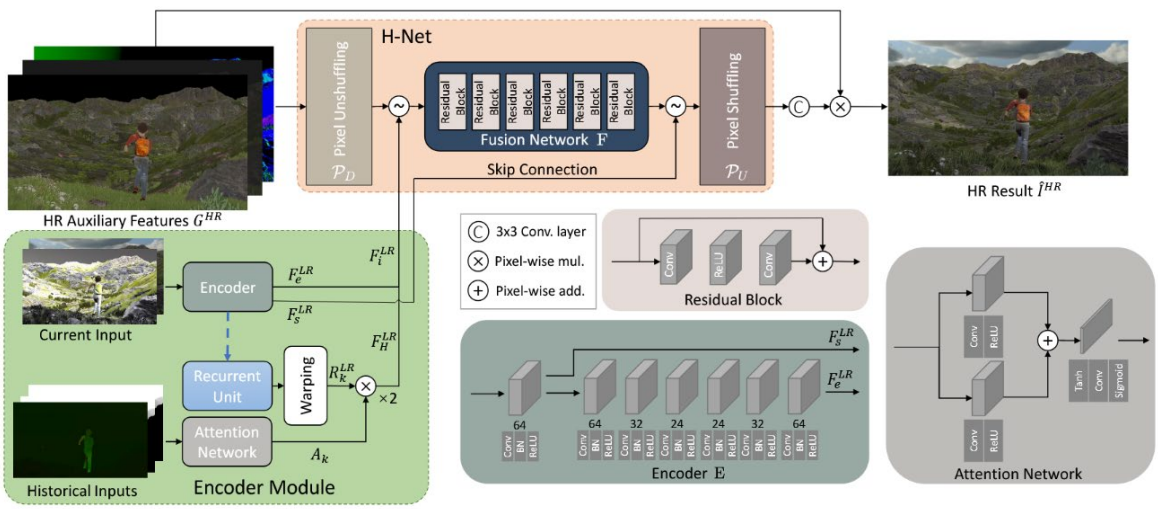


图 4　FuseSR 的整体流程[8]

进一步，Zhang 等[9]利用高清空间 G-buffer 在空域信息上的优势，提出一个支持任意倍率超分的方法 ArbSR，并指出渲染帧颜色与 G-buffer 在傅里叶空间具有相似的频谱分布，其整体流程如图 5 所示。为了准确地提取频谱信息，将渲染内容分解为高频部分和低频部分：高频部分在高清空间重建高频特征编码，可以恢复图像细节且不引入明显鬼影；低频部分使用残差学习方式处理，保持几何结构等低频信息不受高频信号的干扰。为了更好地处理高频细节，该方法还使用空域掩模向网络强调可能出现高频信息的区域。在 ArbSR 的基础上，Zhang 等[20]还设计了一种融合高清 G-buffer 特征的超分方法，其整体流程如图 6 所示。与之前的超分工作相比，该方法不需要考虑历史样本的失效问题，可以处理任意场景内容剧烈变化的场景。

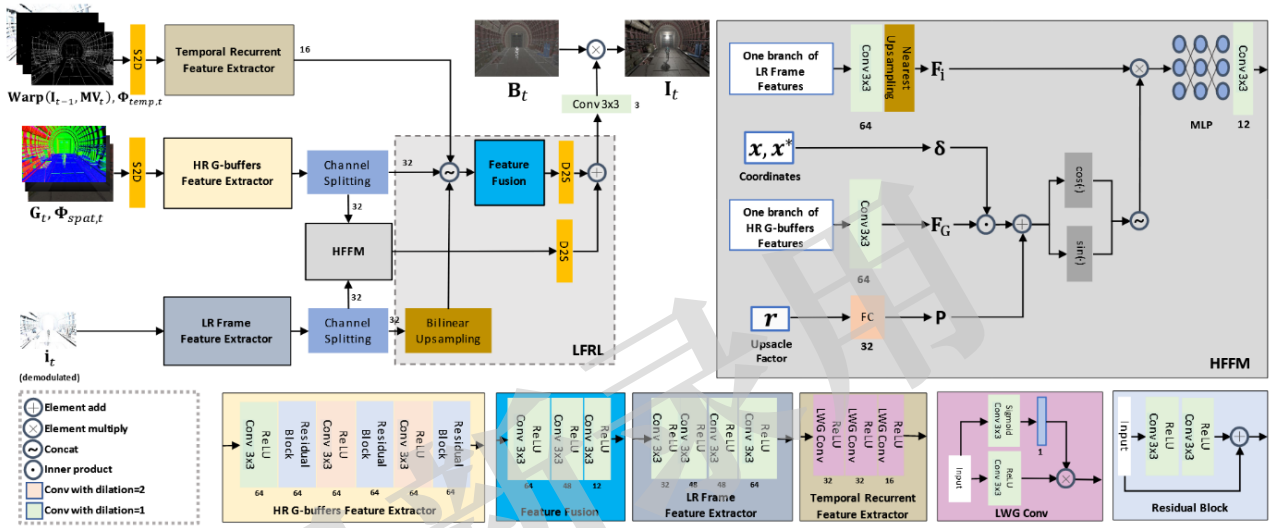


图 5　ArbSR 的整体流程[9]

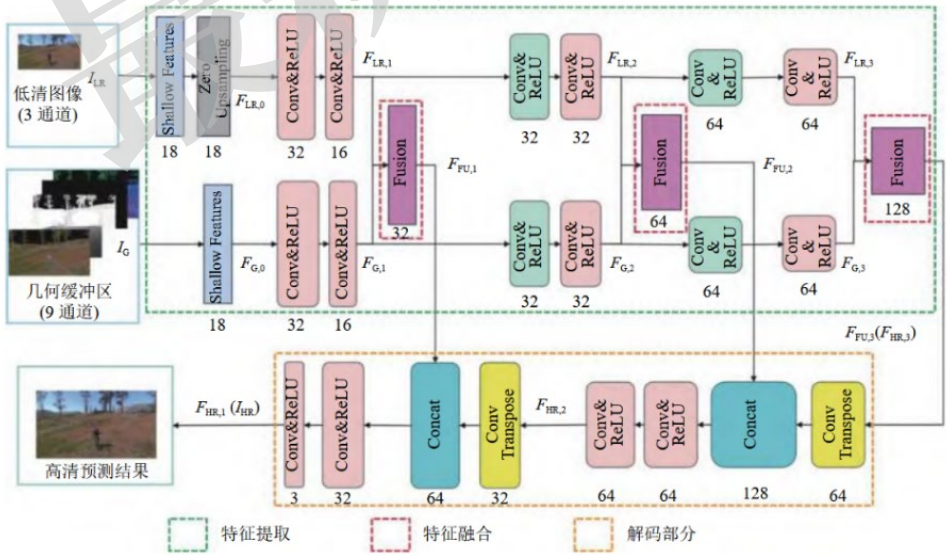


图 6　多尺度特征融合超分的整体流程[20]

在时域方面，多数方法由于使用了高清空间的 G-buffer，因此重点关注利用循环结构维持超分结果时序上的稳定性。FuseSR 将历史 2 帧的低清解调制结果和 G-buffer 作为当前帧的网络输入数据[8]；此外，ArbSR[9]还使用时域掩模作为显式强调；而 Subpixel Sampling Reconstruction[18]则以丰富样本为目的利用低清 G-buffer 的时域信息，提出一种新的蒙特卡罗采样策略，通过将每个像素的采样数在时序上进行分摊来加速采样过程，并设计了一个降噪模块用于重建完整的渲染结果，其网络模块如图 7 所示。降噪模块使用一个循环结构从 G-buffer 和带噪声的渲染帧累计特征，再经过一个 U-Net 结构的网络进行重建；为了保持时序稳定性，重建结果被池化到不同分辨率，在降噪下一帧时拼接在 U-Net 的对应编码层中。

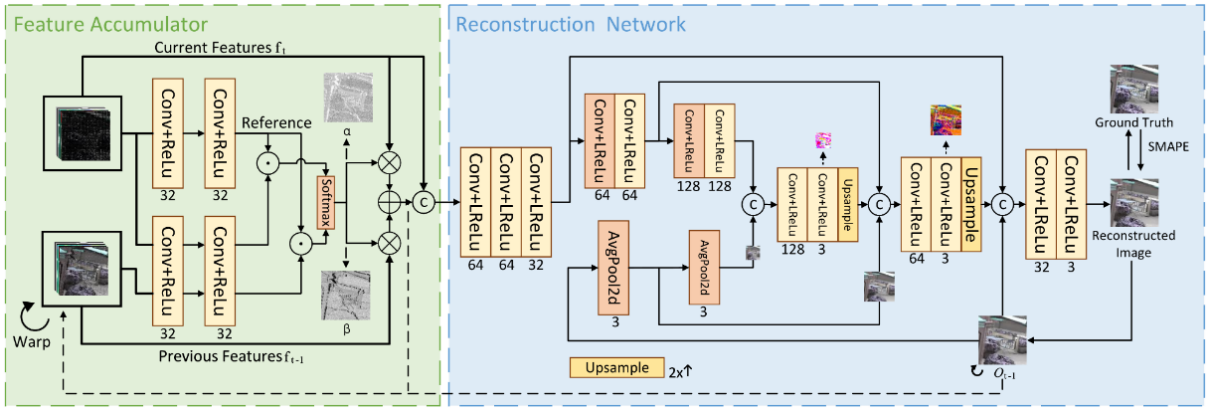


图 7 Subpixel Sampling Reconstruction 方法[18]中的网络模块

2.2 不依赖 G-buffer 的实时渲染超分方法

不依赖 G-buffer 的实时超分方法通常用于发掘渲染帧的时序相关性，在使用 MV 和深度的基础上对时序样本进行提取和筛选，每帧需要的数据量更小，且该方法更容易移植到其他渲染管线。

Xiao 等[16]较早地利用神经网络提取渲染帧的时域信息，提出 NSRR 方法，其整体流程如图 8 所示，其中，当前帧直接零上采样后参与权重评估。为了更好地利用 MV 记录的次像素信息，NSRR 在重投影之前使用“零上采样”处理低清帧，即将低清空间的像素投影到高清空间对应位置，高清空间的其他位置填充零值；零上采样的计算量和采样数少于双线性上采样，在向网络提供样本有效性的同时保留了高频信息，可以得到更好的超分效果。可以看出，使用时序样本重建高清结果时，NSRR 使用一个权重评估网络处理历史帧和当前帧的颜色与深度特征；经过权重调整的历史帧特征和当前帧特征经过一个 U-Net 完成重建。Yang 等[7]面向移动端设计 MNSS，使用了更少的历史帧，减少的时域样本则通过特殊设计的 jitter 模式来弥补，其整体流程如图 9 所示。为了进一步优化在移动端上的表现，MNSS 使用经过后处理量化的网络，并且运行在移动端的神经网络加速硬件上。

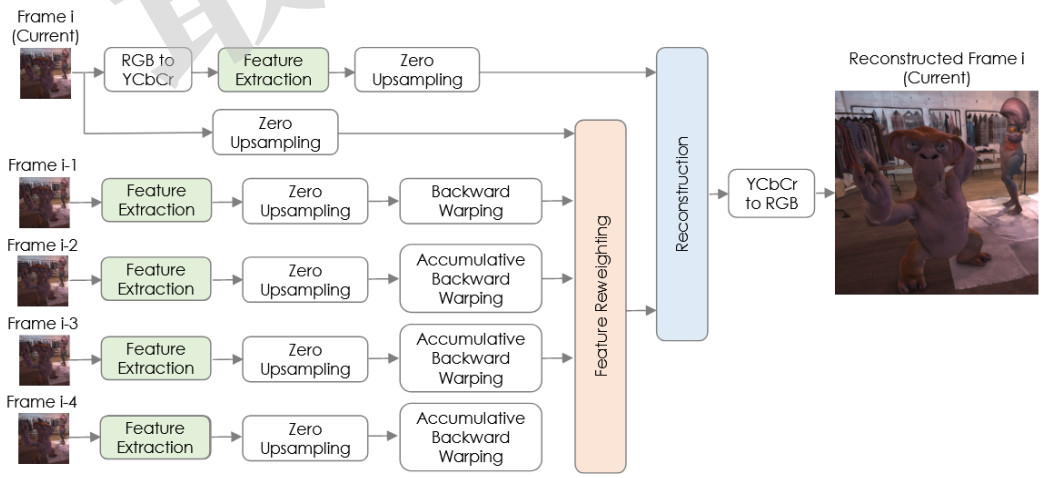


图 8 NSRR 的整体流程[16]

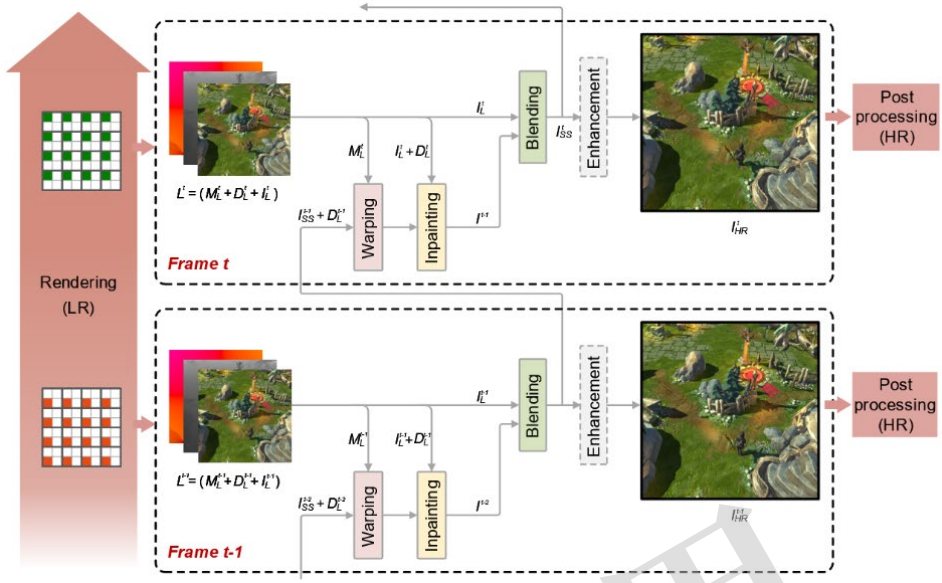


图 9　MNSS 的整体流程[7]

Guo 等[17]基于显式分类的思路利用时域和空域信息，使用一个分类网络从鬼影和走样 2 个角度标记渲染帧的像素，混合网络根据 2 种标记确定时序样本的可靠性并输出混合权重指导混合算法的执行，其整体流程如图 10 所示。

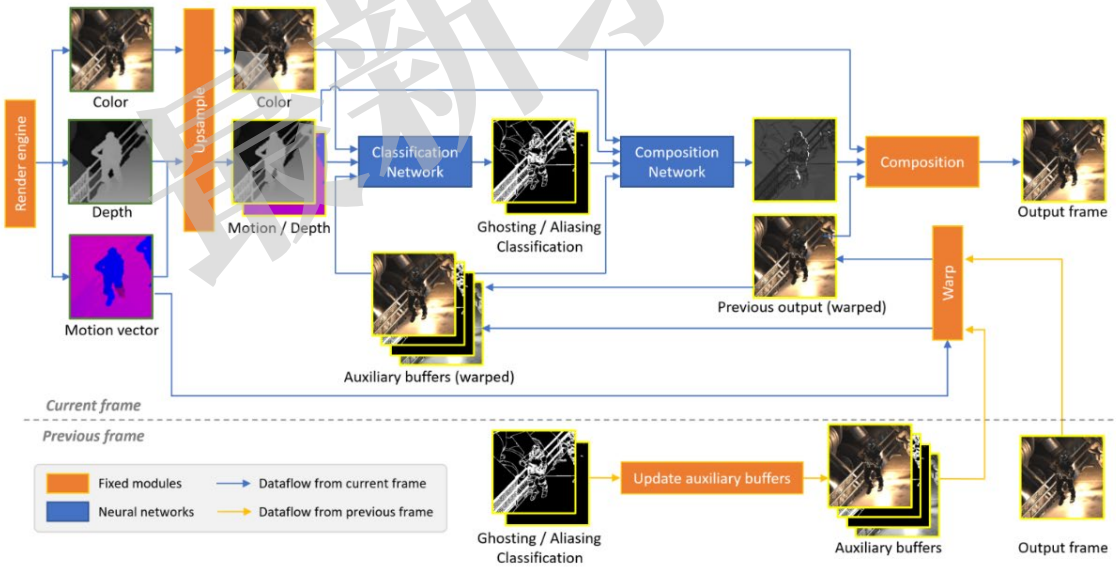


图 10　基于分类器超分的整体流程[17]

在细分领域，也有可以进一步提高实时渲染超分质量或性能的方法。针对游戏渲染，Mercier 等[19]使用负的 mipmap 偏移量渲染低清画面，可以保留较多的高频贴图信息，同时使用计算量更小的循环结构卷积神经网络得到了优于 NSRR 的结果；Ye 等[21]在虚拟现实（virtual reality，VR）领域利用视觉对注视点周围图像更敏感的特点，提出一个空域上异构的超分方法，其整体流程如图 11 所示。该方法将低清颜色帧分块，根据亮度和对比度为每个图块估计一个目标超分质量；不同目标质量的图块由不同深度的网络处理，最后组装成完整的超分结果。与整个渲染帧统一处理相比，该方法的计算量更小。

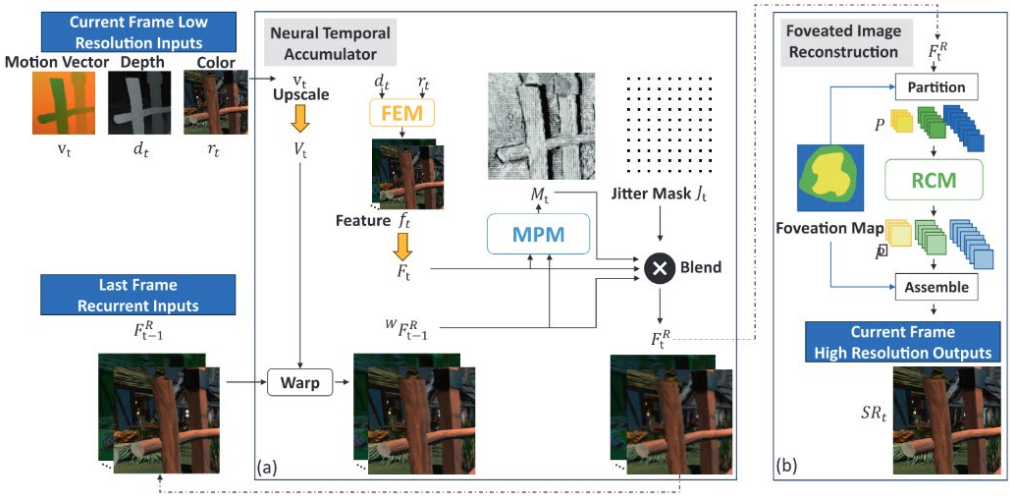


图 11 VR 中基于注视点超分的整体流程[21]

2.3 工业界实时渲染超分方法

在工业界，应用比较广泛的实时渲染超分方法主要有 DLSS[2]和 FSR[1]，它们都是经过长时间迭代的系列方法。本节重点讨论其中的超分辨率部分。

DLSS 给出的方法一直以神经网络为中心。DLSS1.0 使用卷积神经网络进行空域超分，需要对每款游戏进行针对性训练，而 2.0 及之后的 DLSS 实现了神经网络的跨游戏通用，在超分辨率方面也转向空域和时域联合的方法[22]；时序复用方式从 MV 重投影一直发展到 MV 结合光流的重投影[23]。NVIDIA 利用显卡运行游戏时，通过比较空闲的张量核心加速神经网络的推理，并结合了多种优化策略，因此 DLSS 不仅能运行在 NVIDIA 的 RTX 系列显卡上，还对显卡的代数有所要求。

与 DLSS 的闭源不同，FSR 直到第三代一直是开源算法[24]。FSR1 是两阶段的空域超分方法[25]，首先使用边缘自适应空间上采样分析低清颜色帧，确认在高清空间的重建方式；然后使用健壮对比度自适应锐化重建渲染帧细节。FSR2 也采用空域和时域联合的思路[26]，使用 MV 累计时序样本，根据低清的颜色和深度判断样本可用性；针对没有深度信息的半透明和烟雾等效果，FSR2 允许用户提供一个反应性掩模控制像素受历史样本影响的程度，以缓解拖影问题。由于其开源且完全在图形管线上运行，历史版本的 FSR 也可以迁移到其他厂商的硬件平台。

移动平台厂商也在超分辨率方面有所布局，其中比较有代表性的是高通发布的骁龙游戏超分辨率（snapdragon game super resolution，SGSR）[27]。与 PC 平台的商用方法类似，第一代 SGSR 也是空域上采样方法，超分由一个类 Lanczos 滤波器完成，然后使用一个自适应的锐化滤波器增强细节，整个流程可以在一次绘制中完成。SGSR2 则需要多次绘制，首先基于 MV 和深度生成历史样本的可用性阈值；然后使用 Lanczos 插值上采样，与历史帧的可用样本混合得到超分结果。得益于高通的针对性优化，SGSR 在移动端具有非常好的性能表现。

3 实时渲染插帧方法

实时渲染超分的目标是在空域上重建渲染帧的内容，插帧的目标则是在时域上进行重建。根据生成帧的时刻，插帧方法分为内插和外插 2 类。内插方法在第 $t+1$ 帧渲染完成后，在第 t 帧和第 $t+1$ 帧之间插入生成的帧；而外插方法在第 t 帧渲染完成后插入生成的帧。由于未来帧提供的可见性等信息，内插容易得到质量更好的生成帧；而外插方法完全基于对场景变化信息的预测，难度较高，但由于不需要等待未来帧的渲染，理论时延低于内插方法，在实时渲染领域收益明显，因此正在成为实时渲染插帧的主流研究方向。

与超分方法对时域信息的使用类似，现有的插帧方法一般需要将已有的渲染帧重投影到目标生成帧的时刻，为后续的重建和修复提供样本。当前，已有的多数方法使用到目标时刻的 MV 作为参考，但 MV 仍然需要经过渲染得到，意味着渲染管线在目标时刻仍然需要工作，而对物体的运动模拟则一直在目标帧率进行；还有部分方法基于历史帧的 MV 估计到目标帧的运动映射，不需要渲染管线进行目标帧相关的计算。虽然目标时刻的 G-buffer 信息对获取高质量的生成帧也非常重要，但同样地，它们的获取需要额外开销，是否使用以及如何使用

目标时刻相关的信息，各种插帧方法提出了不同的见解。在几何效果之外，正确处理着色效果也可以提高插帧的质量。实时渲染插帧方法对目标帧数据的依赖情况、对 2 种主要效果的处理以及优势场景如表 2 所示。

表 2　实时渲染插帧方法对目标帧数据的依赖情况、对 2 种主要效果的处理以及优势场景

方法	依赖的目标帧数据	几何处理	着色处理	优势场景
ExtraNet[10]	G-buffer	屏幕空间运动估计，几何掩模，多个历史帧输入		处理复杂的几何运动
Learnable MV[12]	G-buffer，自定义 buffer	屏幕空间运动估计，G-buffer 特征提取	自定义 buffer 解耦着色效果，网络特征复用，着色变化导入 MV	适用于复杂着色效果
MoFlow[28]	G-buffer，自定义 buffer	屏幕空间运动估计，G-buffer 特征提取	自定义 buffer 解耦着色效果，网络特征复用，MV 引导光流估计	适用于复杂几何运动与着色效果
GFFE[5]	相机参数	世界空间运动估计，背景收集与修补，动静像素掩模	着色矫正网络	适用管线多样，可见性变化处理较好
ExtraSS[11]	G-buffer	屏幕空间运动估计，基于 G-buffer 的双边滤波	基于流的修补网络	可见性变化处理较好
Mob-FGSR[6]		屏幕空间运动估计		开销极低，适用于移动端
PatchEX[29]		屏幕空间运动估计，前后景掩模与可见性掩模	可变形卷积修补网络	着色修复模块开销低
IBSTI[30]		屏幕空间运动估计，基于辅助相机渲染的修补	阴影掩模	插帧效果好，适用于云游戏等 C/S 架构

3.1　依赖 G-buffer 的实时渲染插帧方法

使用目标时刻的 MV 和 G-buffer，既可以通过准确的 MV 来确定几何运动，也可以通过反照率、法向、世界位置等 G-buffer 信息可靠地处理许多难以预测的可见性变化与着色效果运动。

Guo 等[10]开创性地使用目标时刻 G-buffer 指导网络进行目标帧的预测，提出实时渲染的外插帧生成网络 ExtraNet，其结构如图 12 所示。首先将历史帧数据经过 MV 重投影到目标帧，然后在目标帧 G-buffer 数据的参与下使用神经网络对可见性变化、阴影等效果进行预测和修复；为了处理 MV 重投影导致的鬼影和着色效果错误等问题，通过比较重投影前后的 G-buffer 数据计算出可能的无效区域，如图 13 所示，并使用历史帧编码器处理这些时序上的变化。该方法将 G-buffer 充分应用于神经网络训练推理、渲染帧重投影等过程，保证了生成帧的高质量。

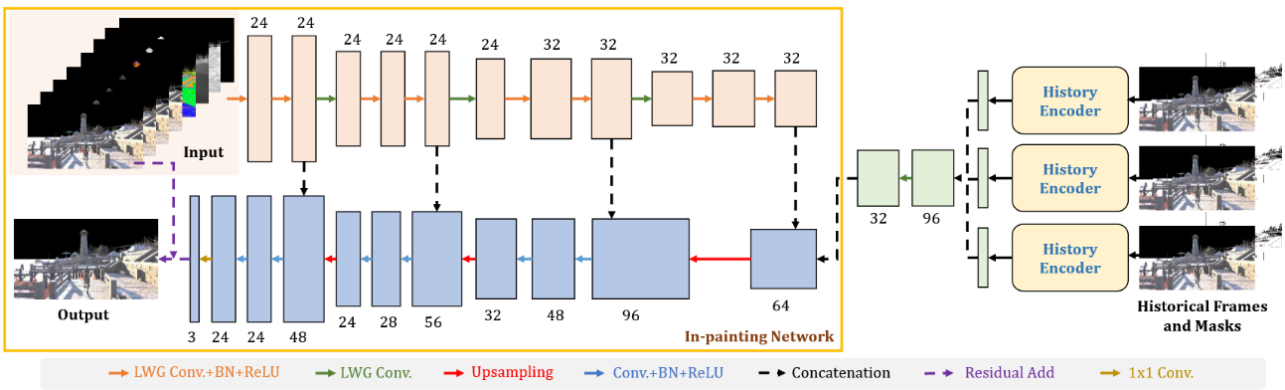
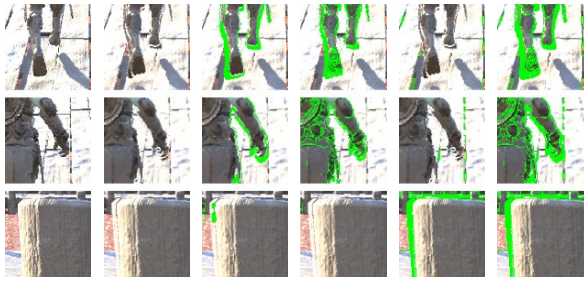


图 12　ExtraNet 的网络结构[10]



a. 渲染效果 b. 重投影效果 c. 模板掩模 d. 法向掩模 e. 位置掩模 f. 所有掩模混合

图 13 ExtraNet 使用的掩模（绿色区域）[10]

覃浩宇等[31]进一步优化了关于 G-buffer 的使用，并引入金字塔光流模块将外插任务分为图像修补与光流重投影 2 部分，在低分辨率下分别处理伪影和着色效果，经过拼接与反卷积得到最终的生成帧。

Wu 等[12]针对仅使用 MV 难以处理透明物体、纹理动画、阴影等不能被 MV 捕捉到的效果的问题，提出一种通过优化 MV 使其可以反映上述效果的方法——Learnable MV，其整体流程如图 14 所示。该方法使用一种双编码器分支的 U-Net 结构：编码器分别处理当前帧和历史帧信息；解码器分支的输入除了当前帧的特征码，还会与经过 MV 重投影的历史帧特征码进行拼接。通过在内部同时处理当前帧、历史帧和重投影关系，能够推理出可学习的 MV 与着色变化的残差，经过组合得到的目标帧可以较好地反映透明物体以及阴影的运动。Learnable MV 充分利用 G-buffer 和渲染帧的时序变化，在准确的几何运动估计上实现了对着色效果的预测。在后续工作中，Wu 等[28]提出 MoFlow，通过自定义 buffer 解耦着色效果，使用 MV 在多个尺度上引导网络输出光流残差，并强化了特征在帧间的循环与帧内的聚合；架构上的优化使得各模块在保证插帧质量的前提下可以使用更简单的算子设计，取得了更快的推理速度。

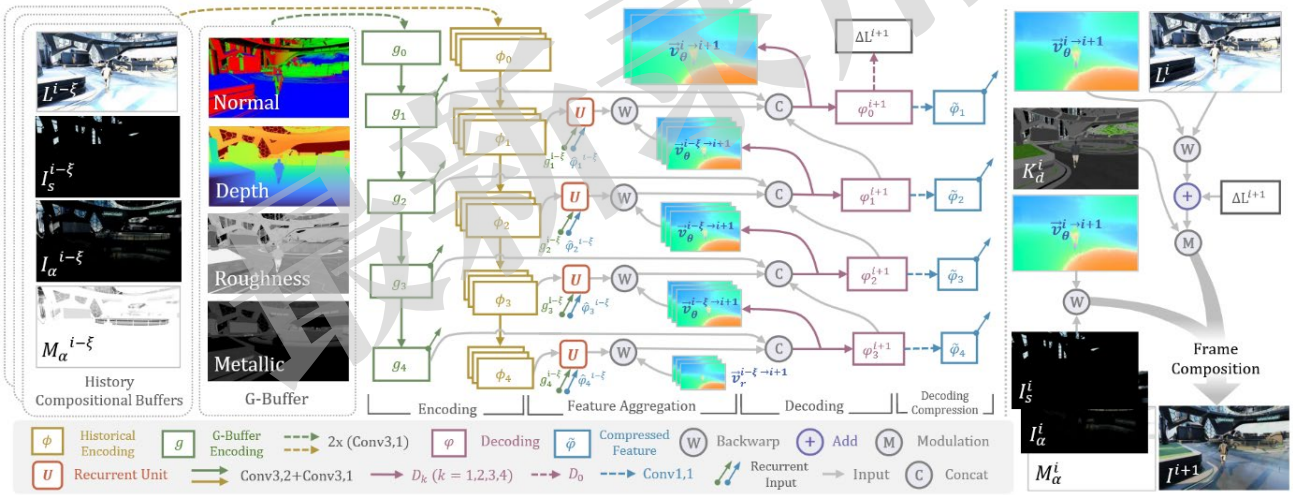


图 14 Learnable MV 的整体流程[12]

3.2 不依赖 G-buffer 的实时渲染插帧方法

依赖 G-buffer 的插帧方法虽然在生成帧效果上有一定优势，但是获取目标帧的 G-buffer 数据需要对渲染管线进行侵入式修改，集成难度较大；在复杂几何场景中，生成 G-buffer 的开销也无法忽略不计。目标帧的 MV 也存在同样的问题。在不依赖目标帧信息的插帧方法中，由于无法取得到目标帧的 MV，已有渲染帧到目标帧的重投影通常完全基于运动估计；而在缺少目标帧 G-buffer 的情况下，遮挡区域的恢复以及着色效果的修复也很难处理。因此，无 G-buffer 插帧方法主要包含运动估计、可见性修复和着色修复。

在估计 MV 方面，VR 领域贡献了 2 个关键的先驱技术：异步时间扭曲（asynchronous timewarp，ATW）[32]和异步空间扭曲（asynchronous spacewarp，ASW）[33]。ATW 是一个轻量级的内插方法，基于 VR 设备的传感器数据对已渲染的图像进行 3D 变换，生成中间帧；ASW 则是更进一步的外插方法，通过分析前一帧到当前帧的头部运动、设备运动等信息合成出下一帧。

Wu 等[5]提出一种在世界空间重投影的运动估计方法 GFFE，整体流程如图 15 所示。首先通过维护像素在世界空间内的运动轨迹，预测它们在生成帧时刻下的位置；然后结合预测的对应时刻相机参数得到目标帧的 MV。作为同期工作，Yang 等[6]提出 Mob-FGSR，考虑在移动端设备上的计算开销，在屏幕空间进行类似的运动估计，以匀加速运动对像素的规矩建模，估计外插或内插时刻的

MV。该方法使用更少的数据减少带宽与运算量，取得了生成帧质量与性能开销之间的平衡。SSAM（screen space aggregated mesh）[34]虽然在相同的空间进行运动估计，但是将运动应用在根据深度梯度生成的屏幕空间网格顶点上，而非每个像素，其预测效果如图 16 所示。可以看出，由于 SSAM 将预测内容限制在静态物体，因此取得了相当高的生成质量。后续的多种方法也沿用了轻量级的屏幕空间运动估计[30-31]。

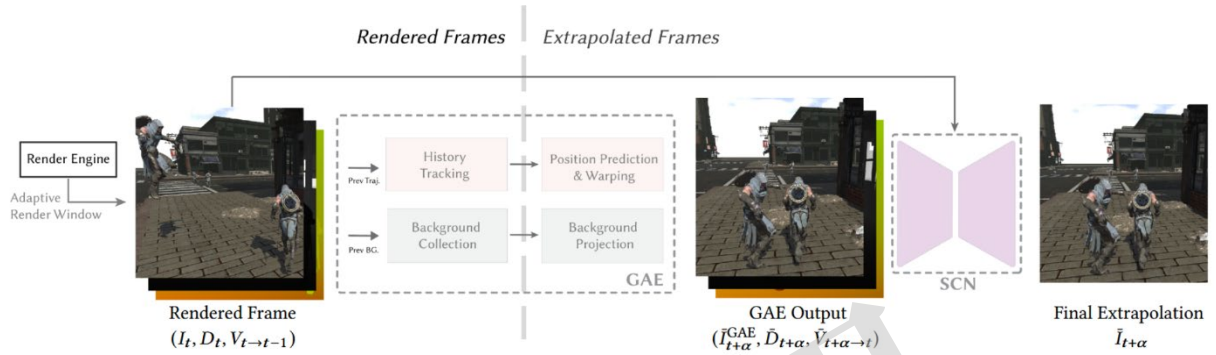
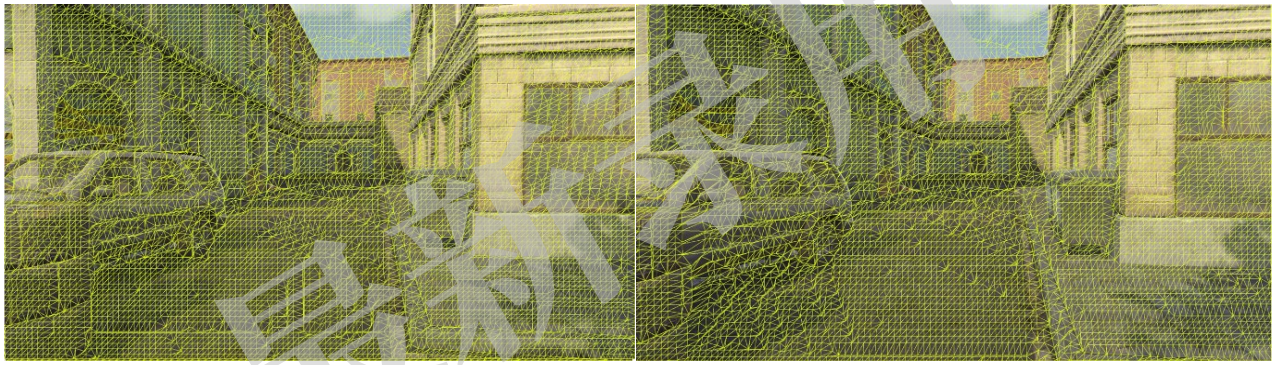


图 15　GFFE 的整体流程[5]



a. SSAM 生成的网格，叠加在渲染的当前颜色帧上　　b. 相机向前移动后的网格与预测的颜色帧

图 16　SSAM 的预测效果[34]

Zhang 等[35]针对屏幕空间运动估计难以处理相机运动导致的视差变化，而世界空间运动估计容易产生像素飞溅或画面撕裂的问题，提出对偶空间运动估计方法，如图 17 所示。静态物体主要受相机运动影响，可以直接在世界空间使用估计的相机矩阵进行投影；动态物体还有自身的运动，因此投影到屏幕空间后需要附加屏幕空间的运动估计。

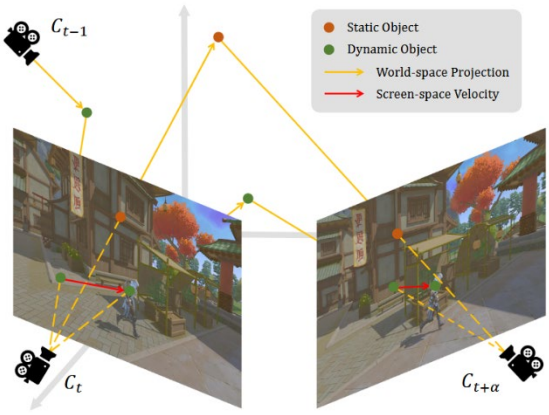


图 17　对偶空间的运动估计[35]

在可见性修复方面，GFFE 收集颜色帧中的背景像素[5]，并通过一个层次优先级的数据结构处理背景中的深度差异，将出现过的背景像素不断累积保存，用于填补鬼影与空洞区域，如图 18 所示。该方法充分地复用历史帧，在修补过程中取得较好的效果。Mob-FGSR[6]在生成内插帧时混合前后帧的重投影结果，从中选择像素填充无效区域，可以较好地处理鬼影与空洞问题；在生成外插帧时则直接使用通过当前帧 MV 寻找到的像素，计算开销小但修复效果有限。Zhang 等[35]从运动估计中预测可见性变化，并使用经过着色修复的颜色帧进行修补，如图 19 所示。Steiner 等[30]提出 IBSTI，在云端使用额外的相机渲染低分辨率的广角画面，客户端则基于额外视角的信息修补无效区域，其整体流程如图 20 所示。虽然利用云端额外的渲染信息能够在保证修补质量的同时不向客户端引入额外的渲染开销，但是数据传输可能导致额外的时延。Dixit 等[29]提出 PatchEX，使用启发式算法标记重投影结果中的无效区域，并利用神经网络进行修

补，取得了较好的修补质量，但是引入了额外的推理开销。

在着色修复部分，GFFE[5]使用一个轻量级的着色矫正网络处理着色变化区域；Mob-FGSR[6]则将这些区域视同鬼影与空洞进行修复；Zhang 等[35]从最近 2 个颜色帧中提取着色变化并传播到目标帧；IBSTI[30]使用在云端渲染的阴影数据矫正客户端的阴影效果，取得了较好的效果。但是，在目标帧信息缺失的前提下处理可见性变化与着色变化，目前的无 G-buffer 插帧方法尚无特别好的方法。

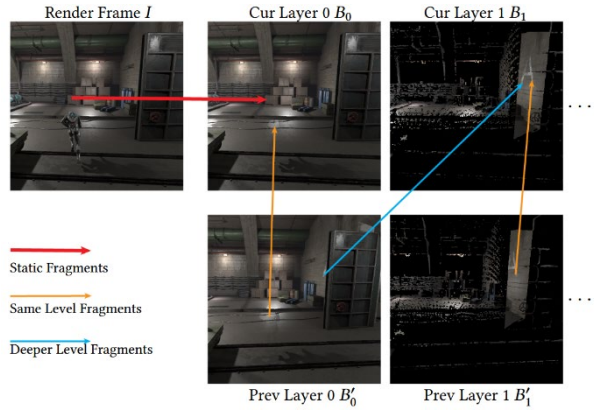


图 18 GFFE 的背景收集方法[5]

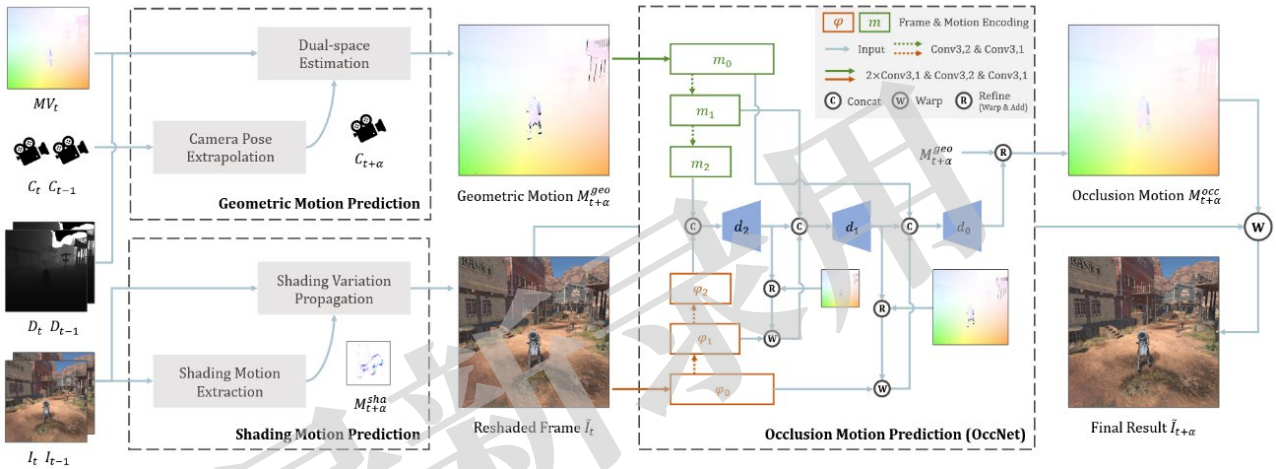


图 19 对偶空间运动估计与可见性修复方法[35]

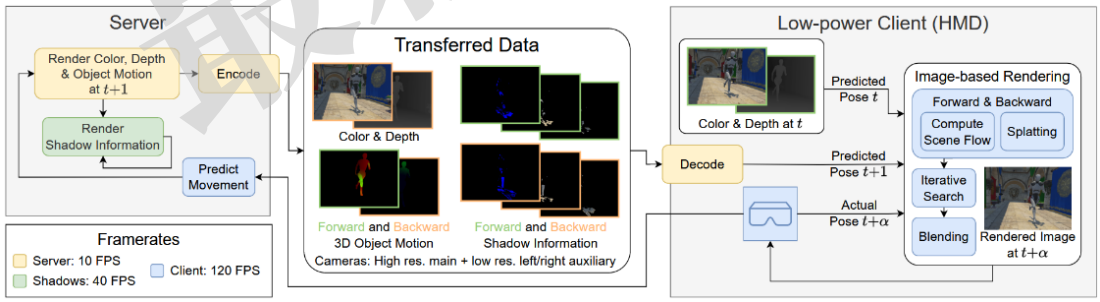


图 20 IBSTI 的整体流程[30]

4 实时渲染超分插帧联合方法

超分与插帧本质上都是利用渲染帧的空域和时域信息，在目标分辨率或者目标时刻重建图像，因此可以很自然地将 2 个任务联合在一起，既增强图像细节，也提高帧率与流畅度，取得重建效果与算法性能之间的平衡。在相同的输出分辨率和帧率下，超分插帧联合的方法可以接受更低的输入分辨率和帧率，在整个实时渲染管线中占用的开销更少；此外，与直接将超分与插帧串联起来相比，共享中间数据、算法模块等结构的联合方法通常可以更高效地复用数据。

当前，超分插帧联合方法主要有 3 种形式：（1）先超分后插帧，即先对低分辨率图像进行空间的超分重建，再对高清结果进行帧插值。这种形式可以为插帧算法提供更多空域信息，有利于提高生成帧质量，与内插方法更加适配；（2）先插帧后超分。这种形式可以减少插帧算法面临的高频干扰，使重建的画面内容有更准确的结构，在低清空间的操作也使得带宽压力与计算压力相对较小，适用于低性能平台；（3）融合策略。采用统一的算法结构或数据流在时空域同时进行重建，避免重复计算并提升整体效率。

4.1 学术界超分插帧联合方法

在使用目标帧信息辅助超分插帧的方法中，比

较有代表性的是 ExtraSS[11]，其整体流程如图 21 所示。受到 ExtraNet 的启发，ExtraSS 使用目标时刻的 G-buffer 信息指导重投影过程中对当前时刻颜色帧的采样，通过混合大面积的像素样本很好地处理了可见性变化导致的鬼影问题；同时，使用一个基于流的轻量级网络处理着色变化，根据低清的历史颜色帧与 G-buffer 预测目标帧的阴影、高光等效果。ExtraSS 的超分基于 U-Net 架构的网络完成，但对渲染帧与生成帧使用不同的编码器，主要区别在于后者的输入包含了经过着色修正的低清目标帧与额外的 G-buffer；双编码器的设计可以调整网络对于不同来源帧的侧重，针对性区分时域-空域超分与时域-空域插帧 2 个任务，减少冗余的输入。

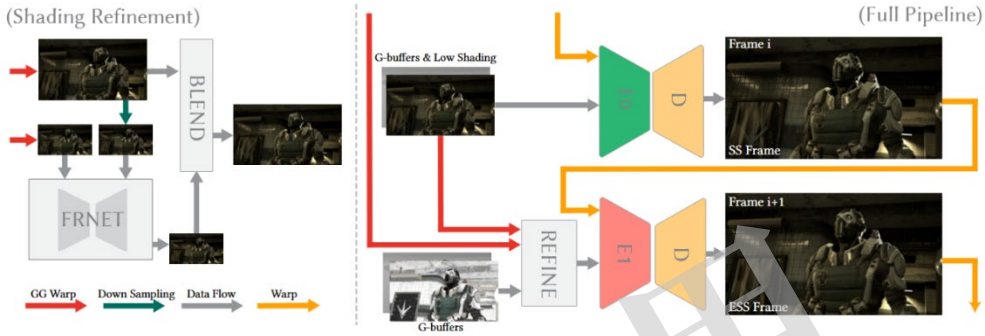
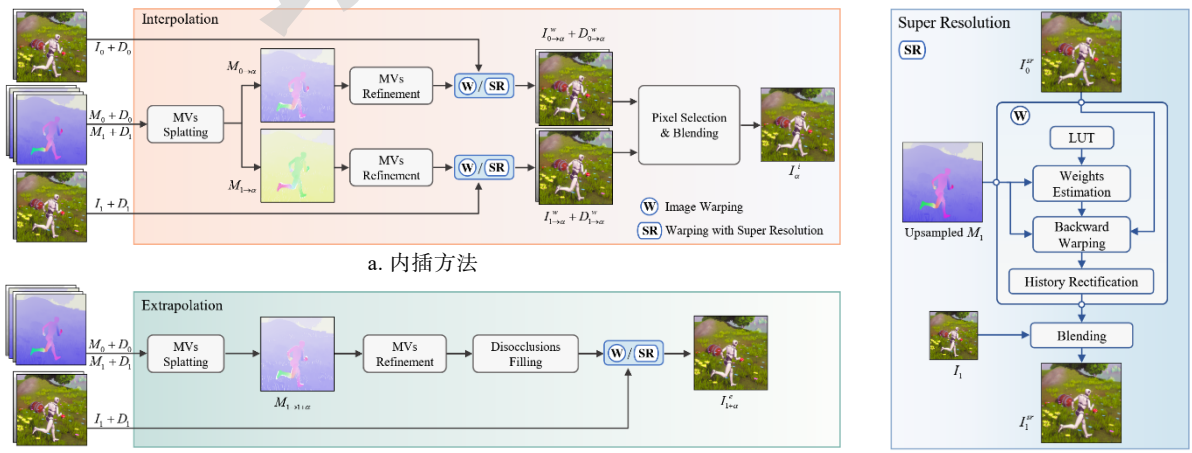


图 21　ExtraSS 的整体流程[11]

Mob-FGSR[6]则是不需要目标帧信息的、面向移动端的超分插帧联合方法，其插帧流程如图 22 所示。在插帧部分，通过屏幕空间的运动估计和遮挡区域的 MV 复用，Mob-FGSR 可以生成历史帧到目标帧的像素映射，用于将已有颜色帧重投影得到低清的目标帧；在超分部分，Mob-FGSR 使用一个多层感知机预测已有高清颜色帧的采样权重，配合像素映射将颜色帧重投影到目标时刻，与低清的目标帧混合后得到高清插帧结果。这个数据驱动的重投影操作能够以更小的计算量获得媲美三线性插值的效果。为了加速推理过程，对多层感知机的查询结果记录为一张查找表（lookup table，LUT），重投影时根据采样位置可以直接获取采样权重。在性能方面，插帧阶段通过运动估计得到的 MV 可以在超分阶段复用，节约了计算开销，整个过程中强调低计算开销的算法细节与先插帧后超分的低带宽框架相适配，对移动端等低性能设备十分友好。



a. 内插方法

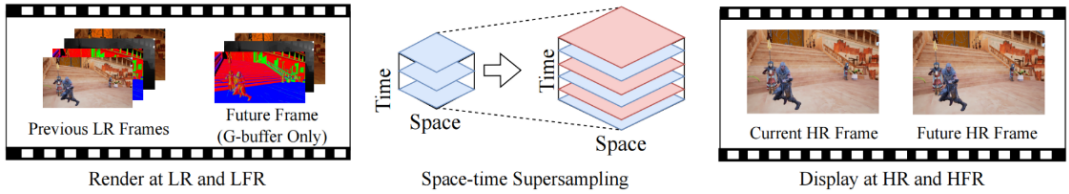
b. 外插方法

c. 内插与外插共用的超分方法

图 22　Mob-FGSR 的插帧与超分流程[6]

伴随神经网络技术在计算机图形学领域的广泛应用，不对超分与插帧任务进行区分，而使用统一的网络同时进行时域和空域超采样的方法应运而生。He 等[36]强调帧外插与超采样之间共享上下文和机制的重要性，构建了一个针对实时渲染的统一时空超采样（space-time supersampling，STSS）框架。分离的超分插帧框架与统一 STSS 的对比如图 23 所示。可以看出，STSS 的输入更加统一，网络结构的共享程度更高。STSS 以 U-Net 架构为网络骨架，根据当前时刻的 G-buffer 和多个重投影到当前时刻的低清颜色帧，输出高清的生成帧或渲染帧，其整体流程如图 24 所示。对于低分辨率渲染

中的走样以及重投影中的空洞、鬼影，该方法将其归类为采样不足导致需要重新着色的区域，通过随机掩模和重投影掩模将 2 种区域一起标记，使用一个重着色模块进行处理。G-buffer 提供的场景先验通过局部注意力机制处理，可以被重着色模块用于识别并聚合具有相似特征的时域和空域样本，从而有效地修复空洞和走样区域。



a. 整体过程

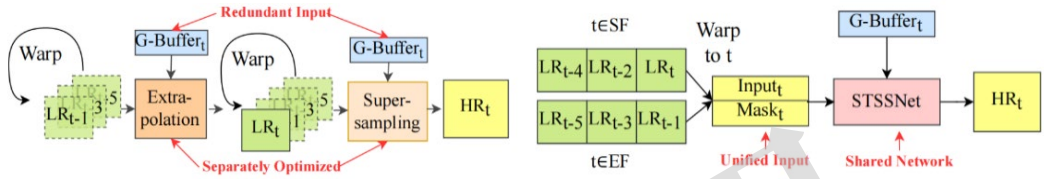


图 23 2 种框架的对比[36]

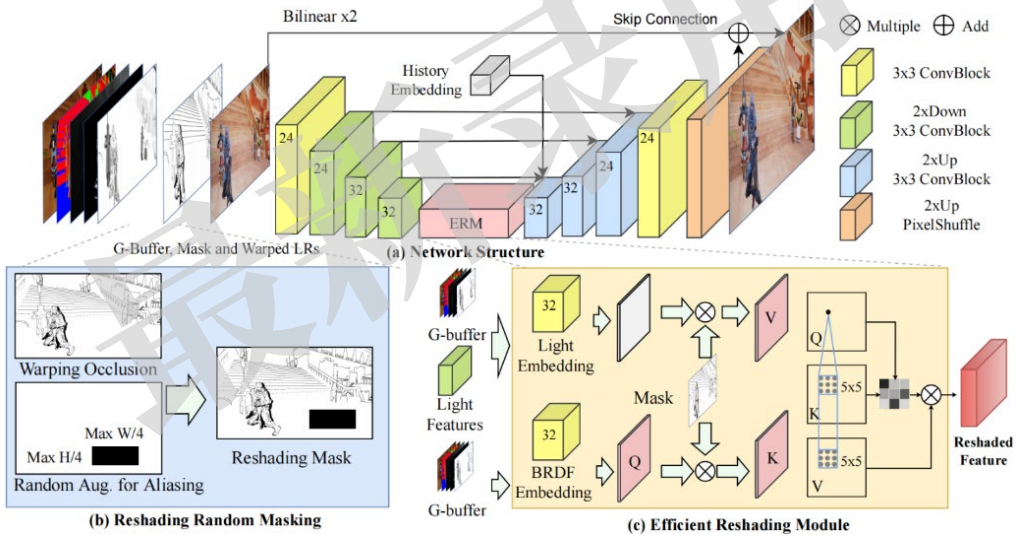


图 24 STSS 的整体流程[36]

作为同期工作，Qin 等[37]同样对统一的超分插帧联合方法进行探索，并提出 FASSET，整体流程如图 25 所示。与 STSS 不同，FASSET 结合隐式神经表示生成任意分辨率结果，通过隐式神经表示模块将一帧的空域信号定义为一个连续函数，用于将任意空间坐标映射到像素值，生成任意目标分辨率的图像；同时，由于隐式神经表示的连续性与网络权重的跨帧共享，连续的输出结果可以保持良好的时序稳定。

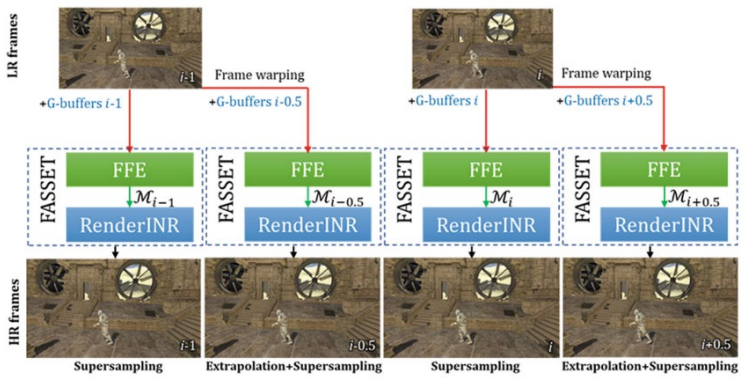


图 25 FASSET 的整体流程[37]

4.2　工业界超分插帧联合方法

目前，商用的超分插帧联合方法在 PC 端应用最广泛，为了充分利用 PC 提供的带宽与算力，同时便于迭代和维护，这些方法多采用先超分后内插的解耦形式实现。

DLSS3.0 同样利用 NVIDIA GPU 的张量核心执行深度学习任务，并正式加入了内插帧技术[23]，首先通过超分网络将 1 080P 的低清颜色帧超分到最高 4K 分辨率，然后使用硬化的光流加速器计算相邻 2 个渲染帧的光流场，高清颜色帧、光流场与 MV、场景深度等游戏引擎数据经过卷积神经网络处理后得到高清的内插帧，其整体流程如图 26 所示。在 DLSS4.0 中，超分辨率模型使用 Vision Transformer[38]作为基础架构，与卷积神经网络的局部感受野相比，Transformer 架构可以考虑一帧或多帧的全部空域信息[39]；同时，硬化的光流加速器被更高效的神经网络取代，配合更高效的帧生成模型可以支持多帧生成。所有颜色帧的呈现与同步控制则由 CPU 侧转移到 GPU 侧，便于硬件统一管理。

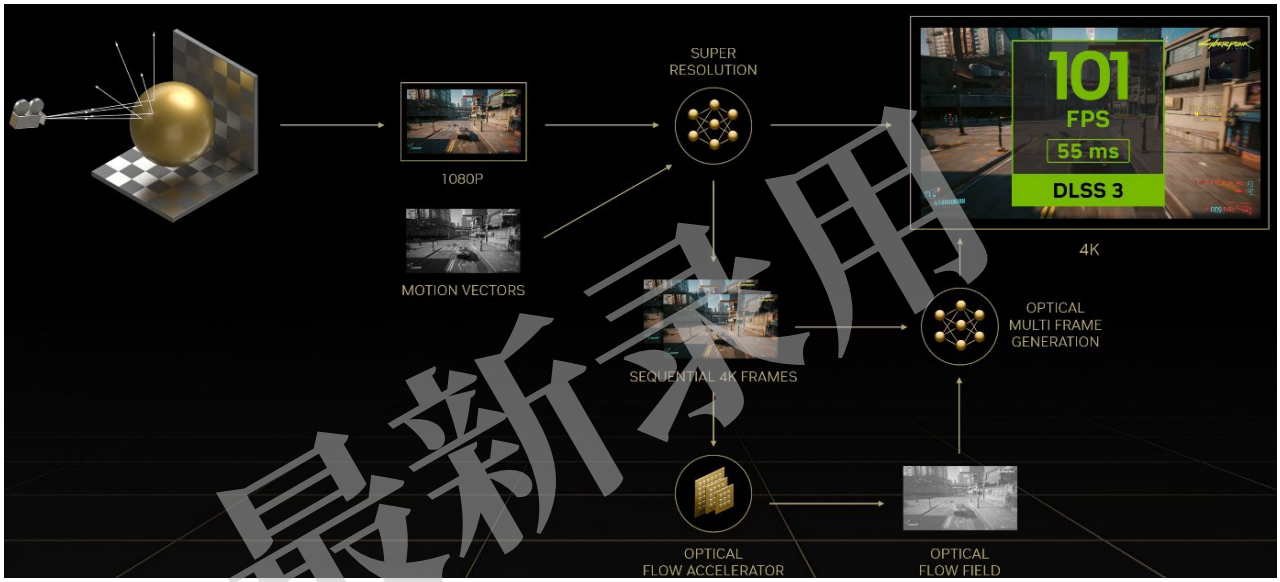


图 26　DLSS3.0 的整体流程[23]

FSR3[40]的设计侧重于硬件兼容性和模块化，它的超分插帧联合方法由时空联合超分、光流计算、帧内插和帧生成交换链 4 个部分组成，每部分都是独立迭代的技术，其整体流程如图 27 所示。时空联合超分根据颜色帧的亮度变化估计每个像素样本的可用性，使用基于规则的方法生成高清帧；光流计算则根据相邻颜色帧的亮度匹配生成光流场；帧生成部分根据 MV 和光流场混合高清的前后颜色帧得到高清内插帧；高清的渲染帧与生成帧则通过交换链和帧同步呈现给显示器。FSR4 使用机器学习辅助超分过程，对鬼影、细小结构、粒子等效果的处理明显优于上一代，但是也转向为发布预编译链接库的闭源形式[41]。

Intel 在超分插帧领域也有相当成熟的探索，所提出的 XeSS2[42]在超分与插帧流程中都使用神经网络辅助，采用先超分后插帧的整体流程，如图 28 所示。在超分阶段，历史帧经过重投影后，与上采样的当前帧颜色和 MV 一起经过网络混合得到高清的当前帧；在插帧阶段，高清的前后帧经过 MV 重投影与神经网络辅助的光流重投影，再经由另一个网络混合后得到高清内插帧。为了兼容更多硬件，XeSS2 提供了 2 种硬件路径：一种利用 Intel Arc 显卡内置的 XMX AI 引擎以获得最佳性能和效果，另一种基于 DP4a 指令集的模式可以兼容更多 GPU 设备[43]。与 NVIDIA 的路径类似，Intel 也将在 XeSS3 中应用多帧生成技术[44]。

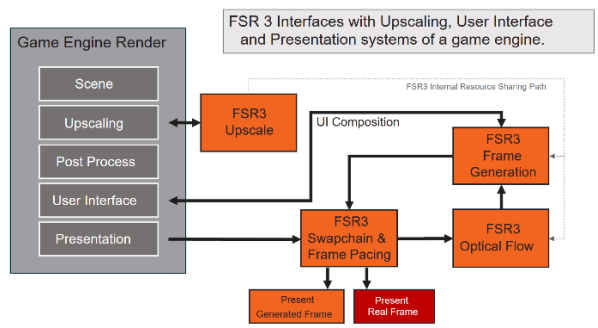
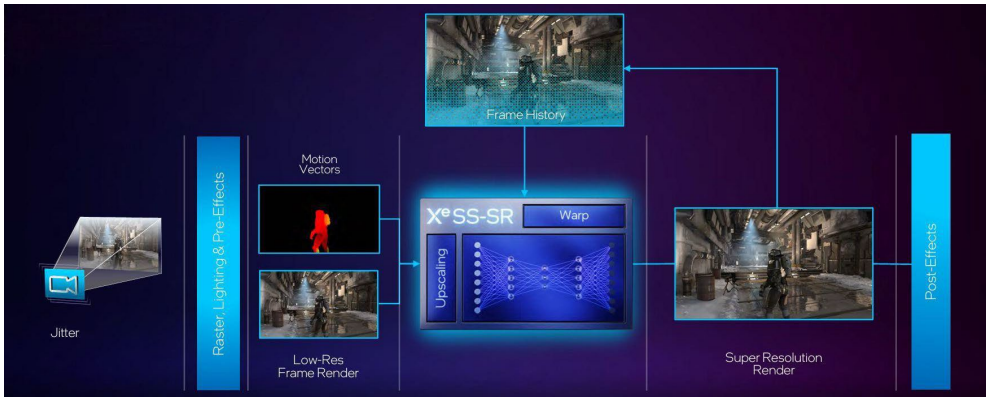
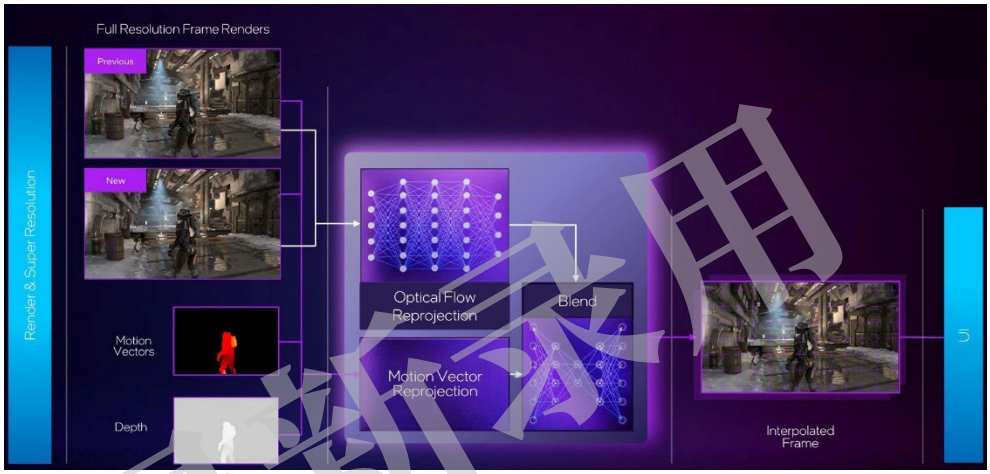


图 27　FSR3 的整体流程[40]



a. XeSS2 超分



b. XeSS2 插帧

图 28 XeSS2 的超分与插帧流程[42]

近年来，移动平台也开始布局专有的超分插帧联合方法。MetalFX[45]提供了基于 Apple Silicon 的超分插帧框架，采用与 DLSS 和 FSR 类似的先超分后插帧流程；G-FRC+方法[46]将超分与内插流程运行在 NPU 硬件上缓解 GPU 的压力；vivo[47]则通过外置独立显示芯片的方式实现移动端超分插帧的并行，在一般的屏幕空间超分外还使用逐纹理超分的方式，进一步增强超分质量。

5 总结与展望

对采样不足的信号进行重建本身就是一个不适定的问题，因此超分与插帧方法总会遇到无法完美处理的场景，这也是该领域未来探索的方向。本文将从时空信息处理、整体方法、集成形态和性能要求等方面，讨论超分插帧未来的可能方向。

发掘帧序列的时域和空域相关性，可以最大程度地为超分或插帧方法提供可用信息。目前，已有的方法多采用将 2D 的颜色信号与几何信息结合，从 3D 角度考虑整个场景的空域信息，同时围绕 MV 与光流处理这些信息的时序变化。未来，将从空域信息中识别场景内的每个“物体”而非所有“像素”，可以针对性地进行超分计算，再结合时域信息考虑“物体”的运动状态，做到更准确的运动估计和区域修复。此外，使用的具体帧内容也会影响超分插帧方法的效果，如 specular 可以辅助识别高光区域，而使用 BRDF 解调制可以有效地提高方法对于光照信息的处理质量。

在整体方法上，AI 辅助已经有了广泛的应用，即使在低端平台上也有实用的量化加速或数据驱动方法。AI 在整个方法中的参与程度由浅到深，可以负责对传统方法的结果进行纠错或修复，也可以作为一个高信息密度的模块存储常见的运动模式或采样权重，甚至可以直接生成完整的目标帧。在具体 AI 辅助形式的探索上，Transformer 网络的用例已经逐渐增加[39,48-50]。未来，超分插帧方法与 AI 相关的方向可能集中在设计高质量高性能的 AI 模型、图形管线与 AI 管线的交互与结合，以及对各种 AI 加速方法的探索。

在集成形态方面，超分与插帧的数据和处理形式应尽量与实时渲染管线靠近，如通过纹理传递帧内容、通过计算管线或其他硬件特性进行加速；

GUI、画面扭曲、电影遮幅黑边等画面内容会影响超分与插帧的效果，需要提供对应的处理接口，如将 GUI 分离绘制并叠加在最终结果上、通过额外的映射指导对扭曲效果的正确采样；目标平台对超分与插帧方法也有很大的影响，PC 端已经有多种比较成熟的超分与插帧方法可供参考和比较，相对较高的性能也允许研究人员进行大胆的设计，如优先追求高质量的复杂方法；移动平台的性能与相对封闭的环境天然限制了可以部署的方法，但是作为一个比 PC 更加可控的软硬件环境，有许多以厂商为个体的研究工作，研究内容也集中在超分插帧算法的部分或全部硬件化、专有的超分插帧芯片等偏底层的方向。根据项目具体特性定制超分或插帧方法则是软件厂商的优势，如将插帧局限在静态物体或直接进行光照处理，以得到更好的质量。

实时渲染内容的超分与插帧始终是辅助方法，性能上要以不影响正常渲染为目标，形式上应尽量透明、易于集成，最终的效果应是在同样开销下有帧率或分辨率的提升，或者在原始帧率或分辨率下有更小的开销。因此，正常渲染与超分插帧的并行化也是一个值得探索的方向。

参考文献（References）：

[1] AMD. AMD FidelityFX™ super resolution[OL]. [2025-12-04]. https://www.amd.com/en/products/graphics/technologies/fidelityfx/super-resolution.html

[2] NVIDIA. DLSS 4 technology | NVIDIA[OL]. [2025-12-04]. https://www.nvidia.com/en-us/geforce/technologies/dlss/

[3] Akenine-Möller T, Haines E, Homan N, *et al*. Real-time rendering[M]. 4th ed. Boca Raton: CRC Press, 2018

[4] Li J, Chen Z L, Wu X L, *et al*. Neural super-resolution for real-time rendering with radiance demodulation[C] //Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Los Alamitos: IEEE Computer Society Press, 2024: 4357-4367

[5] Wu S Y, Vembar D, Sochenov A, *et al*. GFFE: G-buffer free frame extrapolation for low-latency real-time rendering[J]. ACM Transactions on Graphics (TOG), 2024, 43(6): Article No.248

[6] Yang S P, Zhu Q C, Zhuge J H, *et al*. Mob-FGSR: frame generation and super resolution for mobile real-time rendering[C] //Proceedings of the ACM SIGGRAPH Conference Papers. New York: ACM Press, 2024: Article No.64

[7] Yang S P, Zhao Y L, Luo Y Z, *et al*. MNSS: neural supersampling framework for real-time rendering on mobile devices[J]. IEEE Transactions on Visualization and Computer Graphics, 2024, 30(7): 4271-4284

[8] Zhong Z H, Zhu J S, Dai Y X, *et al*. FuseSR: super resolution for real-time rendering through efficient multi-resolution fusion[C] //Proceedings of the SIGGRAPH Asia Conference Papers. New York: ACM Press, 2023: Article No.8

[9] Zhang H N, Guo J, Zhang J W, *et al*. Deep Fourier-based arbitrary-scale super-resolution for real-time rendering[C] //Proceedings of the ACM SIGGRAPH Conference Papers. New York: ACM Press, 2024: Article No.65

[10] Guo J, Fu X H, Lin L Q, et al. ExtraNet: real-time extrapolated rendering for low-latency temporal supersampling[J]. ACM Transactions on Graphics (TOG), 2021, 40(6): Article No.278

[11] Wu S Y, Kim S, Zeng Z, *et al*. ExtraSS: a framework for joint spatial super sampling and frame extrapolation[C] //Proceedings of the SIGGRAPH Asia Conference Papers. New York: ACM Press, 2023: Article No.92

[12] Wu Z Z, Zuo C Y, Huo Y C, *et al*. Adaptive recurrent frame prediction with learnable motion vectors[C] //Proceedings of the SIGGRAPH Asia Conference Papers. New York: ACM Press, 2023: Article No.10

[13] Zeng Z, Liu S Q, Yang J L, *et al*. Temporally reliable motion vectors for real-time ray tracing[J]. Computer Graphics Forum, 2021, 40(2): 79-90

[14] Yang L, Liu S Q, Salvi M. A survey of temporal antialiasing techniques[J]. Computer Graphics Forum, 2020, 39(2): 607-621

[15] Herzog R, Eisemann E, Myszkowski K, *et al*. Spatio-temporal upsampling on the GPU[C] //Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. New York: ACM Press, 2010: 91-98

[16] Xiao L, Nouri S, Chapman M, *et al*. Neural supersampling for real-time rendering[J]. ACM Transactions on Graphics (TOG), 2020, 39(4): Article No.142

[17] Guo Y X, Chen G J, Dong Y, *et al*. Classifier guided temporal supersampling for real-time rendering[J]. Computer Graphics Forum, 2022, 41(7): 237-246

[18] Zhang B Y, Yuan H L. High-quality real-time rendering using subpixel sampling reconstruction[C] //Proceedings of the 38th AAAI Conference on Artificial Intelligence. Palo Alto: AAAI Press, 2024: 7006-7014

[19] Mercier A, Erasmus R, Savani Y, *et al.* Efficient neural supersampling on a novel gaming dataset[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. Los Alamitos: IEEE Computer Society Press, 2023: 296-306

[20] Zhang Haonan, Guo Jie, Qin Haoyu, *et al*. Super-resolution method for rendered contents by multi-scale feature fusion with high-resolution geometry buffers[J]. Journal of Software, 2024, 35(6): 3052-3068(in Chinese)
(张浩南, 过洁, 覃浩宇, 等. 高清几何缓存多尺度特征融合的渲染超分方法[J]. 软件学报, 2024, 35(6): 3052-3068)

[21] Ye J N, Meng X X, Guo D Y, *et al*. Neural foveated super-resolution for real-time VR rendering[J]. Computer Animation and Virtual Worlds, 2024, 35(4): Article No.e2287

[22] NVIDIA. NVIDIA DLSS 2.0: a big leap in AI rendering | GeForce news | NVIDIA[OL]. [2025-12-04]. https://www.nvidia.com/en-us/geforce/news/nvidia-dlss-2-0-a-big-leap-in-ai-rendering

[23] Introducing NVIDIA DLSS 3 | GeForce news | NVIDIA[OL]. [2025-11-24]. https://www.nvidia.com/en-us/geforce/news/dlss3-ai-powered-neural-graphics-innovations

[24] AMD. GPUOpen-LibrariesAndSDKs/FidelityFX-SDK: the

main repository for the FidelityFX SDK[OL]. [2025-12-04]. https://github.com/GPUOpen-LibrariesAndSDKs/FidelityFX-SDK

[25] AMD. FidelityFX super resolution 1.2 (FSR1) | GPUOpen manuals[OL]. [2025-12-04]. https://gpuopen.com/manuals/fidelityfx_sdk/techniques/super-resolution-spatial

[26] AMD. FidelityFX super resolution 2.3.4 (FSR2) | GPUOpen manuals[OL]. [2025-12-04]. https://gpuopen.com/manuals/fidelityfx_sdk2/techniques/super-resolution-temporal

[27] Qualcomm. Introducing snapdragon game super resolution 2[OL]. [2025-12-04]. https://www.qualcomm.com/developer/blog/2024/10/introducing-snapdragon-game-super-resolution-2

[28] Wu Z Z, Yuan Z L, Zuo C Y, *et al*. MoFlow: motion-guided flows for recurrent rendered frame prediction[J]. ACM Transactions on Graphics, 2025, 44(2): Article No.22

[29] Dixit A, Sarangi S R. PatchEX: high-quality real-time temporal supersampling through patch-based parallel extrapolation[J]. ACM Transactions on Graphics, 2025, 45(1): Article No.2

[30] Steiner M, Köhler T, Radl L, *et al*. Image-based spatio-temporal interpolation for split rendering[J]. Computer Graphics Forum, 2025, doi: 10.1111/cgf.70215

[31] Qin Haoyu, Guo Jie, Zhang Haonan, *et al*. High-performance lightweight frame extrapolation technique based on optical flow reprojection[J]. Journal of Zhejiang University (Engineering Science), 2025, 59(5): 902-911(in Chinese)
(覃浩宇, 过洁, 张浩南, 等. 基于光流重投影的高性能轻量级帧外插技术[J]. 浙江大学学报(工学版), 2025, 59(5): 902-911)

[32] Meta. Asynchronous timewarp on oculus rift | meta horizon OS developers[OL]. [2025-12-04]. https://developers.meta.com/horizon/blog/asynchronous-timewarp-on-oculus-rift/

[33] Meta. Asynchronous SpaceWarp | meta horizon OS developers[OL]. [2025-12-04]. https://developers.meta.com/horizon/blog/asynchronous-spacewarp/

[34] Tencent Games. 144FPS rendering on mobile: frame prediction in 'arena breakout'[OL]. [2025-12-04]. https://gdcvault.com/play/1034828/144FPS-Rendering-on-Mobile-Frame

[35] Zhang J W, Zhang H N, Zhang W T, *et al*. Decoupled motion prediction for real-time g-buffer free frame extrapolation[C] //Proceedings of the 33rd ACM International Conference on Multimedia. New York: ACM Press, 2025: 6781-6790

[36] He R A, Zhou S L, Sun Y Q, *et al*. Low-latency space-time supersampling for real-time rendering[C] //Proceedings of the 38th AAAI Conference on Artificial Intelligence. Palo Alto: AAAI Press, 2024: 2103-2111

[37] Qin H Y, Zhang H N, Guo J, *et al*. FASSET: frame supersampling and extrapolation using implicit neural representations of rendering contents[C] //Proceedings of the 12th International Conference on Computational Visual Media. Heidelberg: Springer, 2024: 177-196

[38] Dosovitskiy A, Beyer L, Kolesnikov A, *et al*. An image is worth 16x16 words: transformers for image recognition at scale[C] //Proceedings of the 9th International Conference on Learning Representations. San Diego (CA): OpenReview.net, 2021: 1-22

[39] NVIDIA. NVIDIA DLSS 4 introduces multi frame generation & enhancements for all DLSS technologies | GeForce news | NVIDIA[OL]. [2025-12-04]. https://www.nvidia.com/en-us/geforce/news/dlss4-multi-frame-generation-ai-innovations/

[40] AMD. FidelityFX super resolution 3.1.4 (FSR3) - upscaling and frame generation | GPUOpen manuals[OL]. [2025-12-04]. https://gpuopen.com/manuals/fidelityfx_sdk/techniques/super-resolution-interpolation/

[41] AMD. AMD FSR™ upscaling 4.0.3 | GPUOpen manuals[OL]. [2025-12-04]. https://gpuopen.com/manuals/fidelityfx_sdk2/techniques/super-resolution-ml

[42] Intel. Intel® Xe super sampling 2 whitepaper[OL]. [2025-12-04]. https://www.intel.com/content/www/us/en/developer/articles/technical/xess2-whitepaper.html

[43] Intel. Intel® Xe super sampling (XeSS) API developer guide[OL]. [2025-12-04]. https://www.intel.com/content/www/us/en/content-details/774340/intel-xe-super-sampling-xess-api-developer-guide.html

[44] Intel. Intel technology tour 2025 summary - panther lake[OL]. [2025-12-04]. https://www.intel.com/content/www/us/en/content-details/866361/intel-technology-tour-2025-summary-panther-lake.html

[45] Apple. MetalFX | apple developer documentation[OL]. [2025-12-04]. https://developer.apple.com/documentation/metalfx

[46] Yao Yifei. Introduction to Qualcomm game frame interpolation[OL]. [2025-12-04]. https://www.bilibili.com/video/BV1SgWBzYEeQ/ (in Chinese)
(姚轶非. 高通游戏插帧技术介绍[OL]. [2025-12-04]. https://www.bilibili.com/video/BV1SgWBzYEeQ)

[47] vivo. vivo Developers[OL]. [2025-12-04]. https://developers.vivo.com/doc/d/9730c0eb0083469da0cec73a53ad31bd

[48] Liu Wenting, Lu Xinming. Research progress of Transformer based on computer vision[J]. Computer Engineering and Applications, 2022, 58(6): 1-16(in Chinese)
(刘文婷, 卢新明. 基于计算机视觉的 Transformer 研究进展[J]. 计算机工程与应用, 2022, 58(6): 1-16)

[49] Shi Changtong, Shan Hongtao, Zheng Guangyuan, *et al*. Video frame interpolation method based on improved visual Transformer[J]. Application Research of Computers, 2024, 41(4): 1252-1257(in Chinese)
(石昌通, 单鸿涛, 郑光远, 等. 改进视觉Transformer的视频插帧方法[J]. 计算机应用研究, 2024, 41(4): 1252-1257)

[50] Xiong Wei, Xiong Chengyi, Gao Zhirong, *et al*. Image super-resolution with channel-attention-embedded Transformer[J]. Journal of Image and Graphics, 2023, 28(12): 3744-3757(in Chinese)
(熊巍, 熊承义, 高志荣, 等. 通道注意力嵌入的 Transformer 图像超分辨率重构[J]. 中国图象图形学报, 2023, 28(12): 3744-3757)