Федеральное государственное бюджетное образовательное учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт: *ИВТИ* Кафедра: *Вычислительных машин систем и сетей*

Направление подготовки: *09.03.01 Информатика и вычислительная техника*

ОТЧЕТ по практике

Наименование практики: Производственная практика: научно-исследовательская работа

СТУДЕНТ

_____ / _____ /
(подпись) (*Фамилия и инициалы*)

Группа *А-08-19*
(номер учебной группы)

ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ ПО ПРАКТИКЕ

_____
(отлично, хорошо, удовлетворительно, неудовлетворительно, зачтено, не зачтено)

_____ / _____ /
(подпись) (*Фамилия и инициалы члена комиссии*)

_____ / _____ /
(подпись) (*Фамилия и инициалы члена комиссии*)

Москва
2022

Содержание

ГРАФИК ПРОХОЖДЕНИЯ НИР

Номер п/п	Перечень работ в соответствии с заданием	Отметка о выполнении работы (выполнено /не выполнено)
1		
2		
3		
4		
5		

Руководитель практики (от МЭИ) _____ / _____ /

(подпись) (*Фамилия и инициалы*)

1. Введение

В настоящее время для улучшения изображений используют множество разных технологий, одними из которых являются DLSS и FSR. Эти технологии в своей основе являются развитием временного анти-алиасинга. Они используют поток кадров для сравнения и повышения качества картинки при помощи алгоритмов и нейросетей. Уже сейчас эти технологии используются для улучшения графики в компьютерных играх при понижении требовательности к производительности машин.

2. Технология TAA

1.1 Описание

TAA (Temporal anti-aliasing) - это метод пространственного сглаживания для компьютерного видео, который объединяет информацию из прошлых кадров и текущего кадра для удаления ступенчатости в текущем кадре.

1.2. Алгоритм работы

Предположим, что для каждого пикселя перед кадром N было собрано несколько сэмплов (желтые точки) (смотреть рисунок 1.1), которые были усреднены и сохранены в буфере истории как значение одного цвета для пикселя (зеленая точка). Для каждого пикселя в кадре N сопоставляется его центральное положение (оранжевая точка) с предыдущим кадром N − 1 на основе движения сцены и передискретизируется буфер истории в этом месте, чтобы получить исторический цвет для этого пикселя. При передискретизации исторический цвет представляет собой среднее значение ранее накопленных выборок вокруг этой точки. Для текущего кадра N затеняется новый сэмпл (синяя точка) в месте с дрожанием и результат объединяется с цветом истории передискретизации. В результате получается цвет выходного пикселя кадра N, который затем становится историей для кадра N+1.

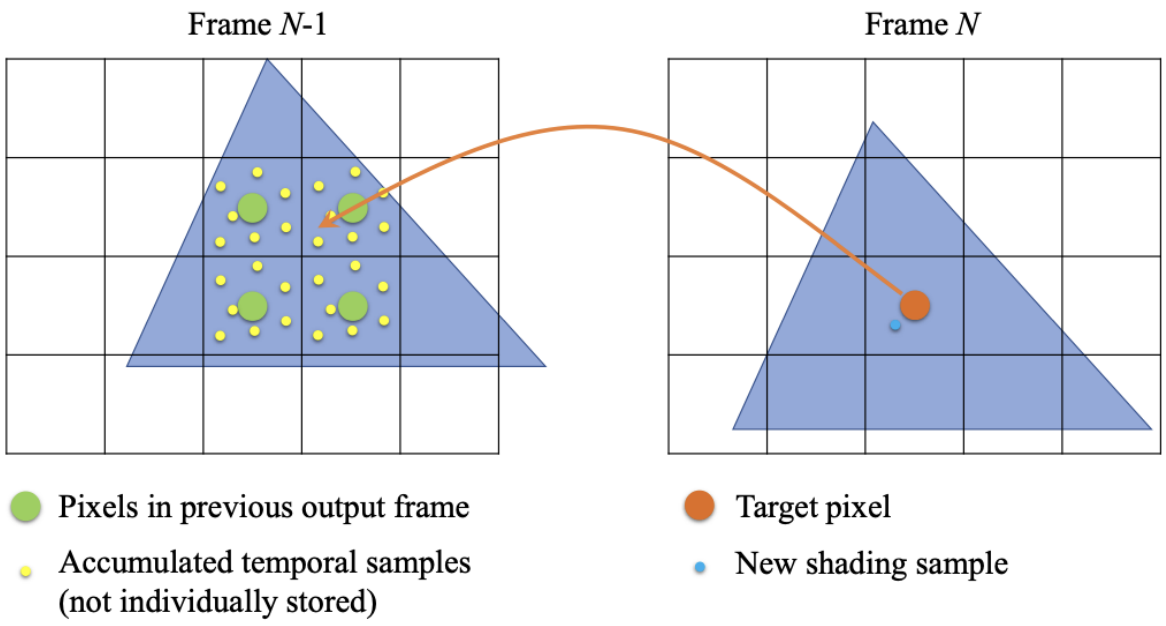


Рис.1.1 Передискретизация [7]

По практическим причинам большинство реализаций TAA рендерят один сэмпл на пиксель в каждом кадре. Обычный подход к созданию разных сэмплов в каждом кадре для всех пикселей заключается в добавлении субпиксельного дрожания (Jitter) области просмотра к матрице проекции камеры. «Дрожание» - это двухмерное смещение пиксельной сетки, а его величины по осям X и Y находятся в диапазоне от 0 до 1. Дрожание для каждого кадра обычно берется из хорошо распределенной последовательности выборок, так что каждый пиксель равномерно покрывается выборками, сгенерированными в нескольких кадрах.

Поскольку объект может появиться или исчезнуть в любой момент, первый сэмпл начала накопления пикселей может начинаться с любого индекса в последовательности. Таким образом, чтобы обеспечить быструю сходимость, идеальная последовательность должна обладать свойством, заключающимся в том, что любая подпоследовательность любой длины должна быть равномерно распределена в пиксельной области. Определенные последовательности с

малым расхождением, такие как последовательности Холтона или Собеля, обладают этим свойством. В зависимости от целевого качества и эффективного количества накопленных образцов длина последовательности часто может быть ограничена относительно небольшим числом. Например, Unreal Engine 4 по умолчанию использует последовательность из 8 сэмплов из последовательности Холтона(2, 3).

При использовании TAA сэмплинг и интегрирование по площади пикселей обеспечивают сглаживание как геометрии, так и текстуры. Поскольку текстуры обычно фильтруются с помощью MIP -текстурирования, они могут быть размыты TAA на выходе. Поэтому обычно к прямому проходу, где сэмплируются текстуры, применяется смещение MIP -текстуры. Это особенно важно для TAAU (TAA Upsampling). Уровень смещения MIP-текстуры должен компенсировать соотношение между эффективной плотностью сэмпла и входной плотностью пикселей. Например, если ожидается, что эффективное количество сэмплов на входной пиксель будет равно 4, то смещение мип-текстуры рассчитывается как $-0.5\log2(4) = -1,0$. На практике иногда предпочтительнее менее агрессивное смещение между этим значением и 0, чтобы не повредить временную стабильность, а также эффективность кэширования текстур.

Временной апсемплинг (TAAU) является естественным расширением TAA за счет дальнейшего снижения эффективной частоты дискретизации с одной выборки на пиксель до доли выборки на пиксель. Этого часто требуют приложения с большой нагрузкой на пиксели, и он становится все более популярным в играх, ориентированных на дисплеи с высоким разрешением. TAAU, по существу, накапливает результаты шейдинга с более низким разрешением и создает изображения с более высоким разрешением, которые

часто содержат больше деталей, чем результаты чистого пространственного повышения дискретизации.

3. Технология DLSS

2.1 Описание

DLSS или Deep Learning Super Sampling - это метод интеллектуального масштабирования Nvidia, который может взять изображение, визуализированное с более низким разрешением, и масштабировать его до с более высокого разрешения, что обеспечивает более высокую производительность, чем собственный рендеринг. Nvidia представила эту технологию в первом поколении видеокарт серии RTX. DLSS - это не просто метод обычного апскейлинга или суперсэмплинга, он использует нейросети для интеллектуального повышения качества изображения, которое было обработано с более низким разрешением, чтобы сохранить качество изображения.

Рис. 2.1 Пример работы DLSS [6]

2.2 Отличия от TAA, MSAA и их производных

Методы апскейлинга и суперсэмплинга также реализуют тот же базовый метод масштабирования, что и DLSS; они берут изображение с более низким разрешением и масштабируют его, чтобы оно соответствовало более высокому разрешению. Отличие DLSS в основном состоит из двух пунктов:

- Качество на выходе: качество изображения на выходе игр с традиционным масштабированием обычно ниже, чем с DLSS. Это связано с тем, что DLSS использует нейросети для расчета и настройки качества изображения, чтобы можно было свести к минимуму разницу между исходным и увеличенным изображением. В традиционных методах апскейлинга такой обработки нет, поэтому качество выходного изображения ниже, чем при традиционном рендеринге и DLSS.
- Ухудшение производительности: суперсэмплинг может улучшать изображение с более низким разрешением, но оно не обеспечивает достаточного повышения производительности, чтобы оправдать потерю качества изображения. DLSS смягчает эту проблему, обеспечивая значительный прирост производительности, сохраняя при этом качество изображения, очень близкое к изначальному.

TAAU используется во многих современных видеоиграх и игровых движках, однако во всех предыдущих реализациях использовалась та или иная форма написанной вручную эвристики для предотвращения временных артефактов, таких как ореолы и мерцание. Одним из примеров этого является фиксация соседства, которая принудительно предотвращает слишком большое отклонение выборок, собранных в предыдущих кадрах, по сравнению с соседними пикселями в более новых кадрах. Это помогает идентифицировать и исправить многие временные артефакты, но преднамеренное удаление

мелких деталей таким образом аналогично применению фильтра размытия, и поэтому конечное изображение может выглядеть размытым при использовании этого метода.

В DLSS 2.0 используется сверточная нейронная сеть с автоматическим кодировщиком, обученная идентифицировать и исправлять временные артефакты, вместо запрограммированных вручную эвристик, как упоминалось выше. В процессе обучения выходное изображение сравнивается с эталонным изображением сверхвысокого качества 16K, созданным в автономном режиме, и разница передается обратно в сеть, чтобы она могла продолжать обучение и улучшать свои результаты. Этот процесс повторяется на суперкомпьютере десятки тысяч раз, пока сеть не будет надежно выводить высококачественные изображения с высоким разрешением.

Из-за этого DLSS 2.0, как правило, лучше разрешает детали, чем другие реализации TAA и TAAU, а также удаляет большинство временных артефактов. Вот почему DLSS 2.0 иногда может создавать более четкое изображение, чем рендеринг с более высоким или даже родным разрешением с использованием традиционного TAA. Однако нет идеальных временных решений, и при использовании DLSS 2.0 в некоторых сценариях по-прежнему видны артефакты.

Рис. 2.2 Краткий алгоритм работы нейронной сети [2]

Поскольку временные артефакты возникают в большинстве художественных стилей и сред в целом одинаково, нейронную сеть, на которой работает DLSS 2.0, не нужно переобучать при использовании в разных играх. Несмотря на это, Nvidia часто поставляет новые второстепенные версии DLSS 2.0 с новыми играми, так что это может указывать на то, что некоторые незначительные оптимизации обучения могут быть выполнены по мере выпуска игр, хотя Nvidia не предоставляет журналы изменений для этих второстепенных изменений, чтобы подтвердить это.

4. Технология FSR

3.1 Описание

FidelityFX Super Resolution (FSR) используется для повышения разрешения входного изображения до более высокого разрешения. Есть две версии FSR с отличительной техникой апскейлинга и качеством изображения. FSR 1 - это пространственный апскейлер, основанный на алгоритме Ланцоша, требующий сглаженного изображения с более низким разрешением, в то время как FSR 2 - это временной апскейлер, основанный на модифицированном алгоритме Ланцоша, требующий сглаженного изображения с более низким разрешением и использующий временные данные (такие как векторы движения и историю кадров), а затем применяет собственный проход сглаживания, который заменяет решение временного сглаживания игры. Пример работы FSR 2 представлен на рис. 3.1.

Рис. 3.1 Сравнение нативного 1920x1080 (слева) и работы FSR2 при изначальном разрешении 1280x600 (справа)

3.2 Алгоритм работы

Благодаря открытой документации по FSR2 можно описать его работу.

Алгоритм FSR2 реализуется в несколько этапов, которые заключаются в следующем:

- Вычислительная пирамида яркости (Compute luminance pyramid)
- Настроить цвет ввода
- Реконструировать и расширить
- Обрезка глубины
- Создание блокировок
- Перепроецировать и накопить
- Надежная контрастная адаптивная резкость (RCAS)

3.2.1 Вычислительная пирамида яркости

Этап вычислительной пирамиды яркости реализован с помощью FidelityFX Single Pass Downsampler (SPD) - оптимизированного метода создания цепочек MIP-текстур с использованием одной отправки вычислительного затенения. Вместо традиционного (полного) пирамидального подхода SPD предоставляет механизм для создания определенного набора уровней MIP-карт для произвольной входной текстуры, а также для выполнения произвольных вычислений с этими данными, когда мы сохраняем их в целевом месте в памяти.

Значение экспозиции - либо из приложения, либо из этапа вычисления пирамиды яркости - используется на этапе настройки входного цвета FSR2, а также на этапе перепроецирования и накопления. [4]

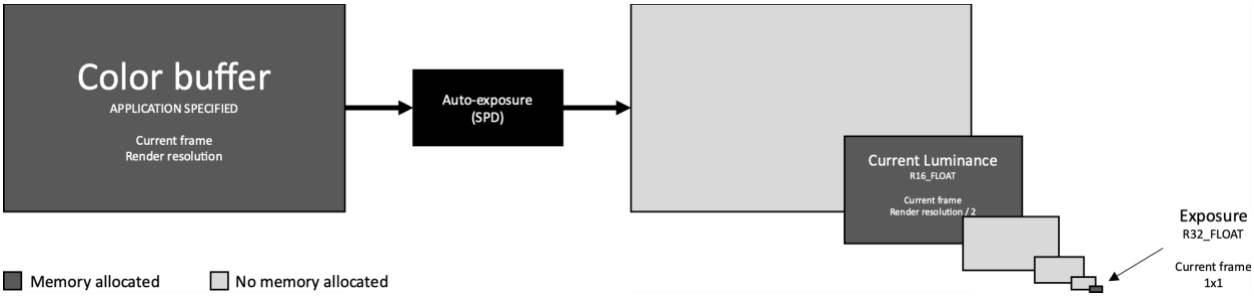


Рис. 3.2 Работа SPD

3.2.2 Настройка цвета ввода

Существует несколько типов корректировок, которые FSR2 выполняет для входных цветов, а именно:

- Входной цвет делится на значение предварительной экспозиции.
- Входной цвет умножается на значение экспозиции.
- Затем выставленный цвет преобразуется в цветовое пространство YCoCg.

Большая часть алгоритма FSR2 работает в цветовом пространстве YCoCg. Цветовая модель YCoCg, также известная как цветовая модель YCgCo, представляет собой цветовое пространство, сформированное путем простого преобразования связанного цветового пространства RGB в значение яркости (обозначаемое как Y) и два значения цветности, называемые цветностью зеленого (Cg) и цветностью оранжевого цвета (Co). Чтобы избежать повторного вычисления преобразований из цветового пространства, используемого приложением, FSR2 реализует специальный этап, который применяет все корректировки к цвету один раз, а затем результаты кэшируются в скорректированную цветовую текстуру, которую затем могут считывать другие проходы. В рамках процесса настройки FSR2 также вычисляет буфер истории яркости.

Поскольку буфер яркости является постоянным (он недоступен для сглаживания или очистки каждого кадра), существует доступ к четырем кадрам истории во время этапа настройки входного цвета на любом кадре. Однако в конце этапа настройки входного цвета значения истории яркости сдвигаются вниз, а это означает, что последующие этапы FSR2 имеют доступ к трем самым последним кадрам яркости (текущий кадр и два кадра перед ним).

В дополнение к уже описанным обязанностям по корректировке цвета, этот этап также несет ответственность за очистку перепроецированного буфера глубины до известного значения, готового к этапу реконструкции и расширения в следующем кадре приложения. Буфер должен быть очищен, так как на этапе реконструкции и расширения он будет заполнен с помощью атомарных операций.

3.2.3 Реконструкция и расширение

Этап реконструкции и расширения использует буфер глубины приложения и векторы движения и создает реконструированный и расширенный буфер глубины для предыдущего кадра вместе с расширенным набором векторов движения в UV-пространстве. Этап выполняется с разрешением рендеринга.

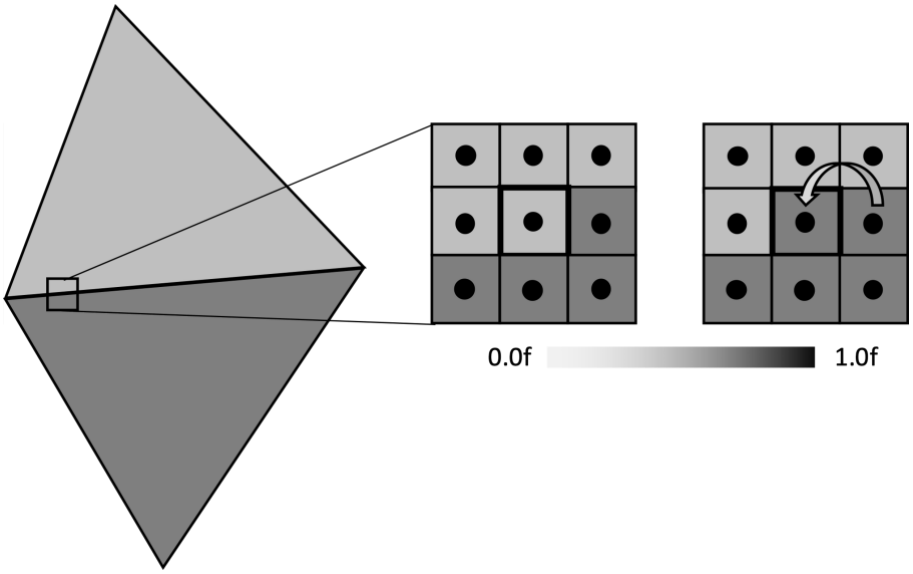


Рис. 3.3 Расширение вектора движения

Первым шагом этапа реконструкции и расширения является вычисление расширенных значений глубины и векторов движения на основе значений глубины и векторов движения приложения для текущего кадра. Расширенные значения глубины и векторы движения подчеркивают края геометрии, которая была визуализирована в буфере глубины.

Это связано с тем, что края геометрии часто будут вводить разрывы в непрерывный ряд значений глубины, а это означает, что по мере расширения значений глубины и векторов движения они будут естественным образом следовать контурам геометрических краев, присутствующих в буфере глубины.

Чтобы вычислить расширенные значения глубины и векторы движения, FSR2 просматривает значения глубины для окрестности 3x3 для каждого пикселя, а затем выбирает значения глубины и векторы движения в той окрестности, где значение глубины находится ближе всего к камере. [4]

С расширенными векторами движения можно перейти ко второй части этапа реконструкции и расширения, который должен оценить положение каждого пикселя в буфере глубины текущего кадра в предыдущем кадре. Это делается путем применения расширенного вектора движения, вычисленного для пикселя, к его значению буфера глубины (Рисунок 3.4). [4]

Поскольку многие пиксели могут перепроецироваться в один и тот же пиксель в предыдущем буфере глубины, используются атомарные операции для разрешения значения ближайшего значения глубины для каждого пикселя. Использование кумулятивных операций для разрешения содержимого предыдущего буфера глубины подразумевает, что реконструированный ресурс буфера глубины всегда должен очищаться до известного значения, что выполняется на этапе «Перепроецировать и накапливать». Это выполняется на кадре N для кадра N + 1.

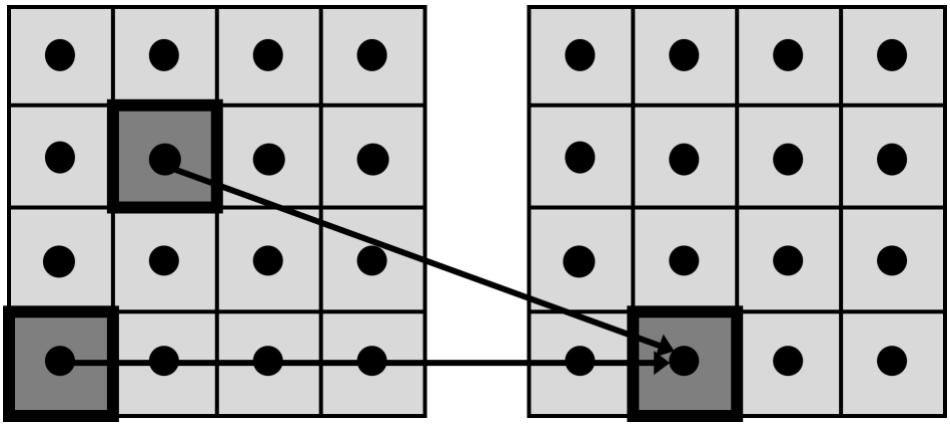


Рис. 3.4 Применение расширенного вектора движения к его значению буфера глубины

3.2.4 Обрезка глубины

Целью этапа обрезки глубины является создание маски, которая указывает незакрытые области текущего кадра.

Этот этап выполняется с разрешением рендеринга.

Чтобы сгенерировать маску дизокклюзии, значение глубины должно быть вычислено для каждого пикселя из положения предыдущей камеры и положения новой камеры. На рисунке 3.5 можно видеть, как камера перемещается из исходного положения (обозначенного P0) в новое положение (обозначенное P1). При этом заштрихованная область за сферой становится непрозрачной, то есть становится видимой для камеры в точке P1 и ранее закрытой с точки зрения точки P0. [4]

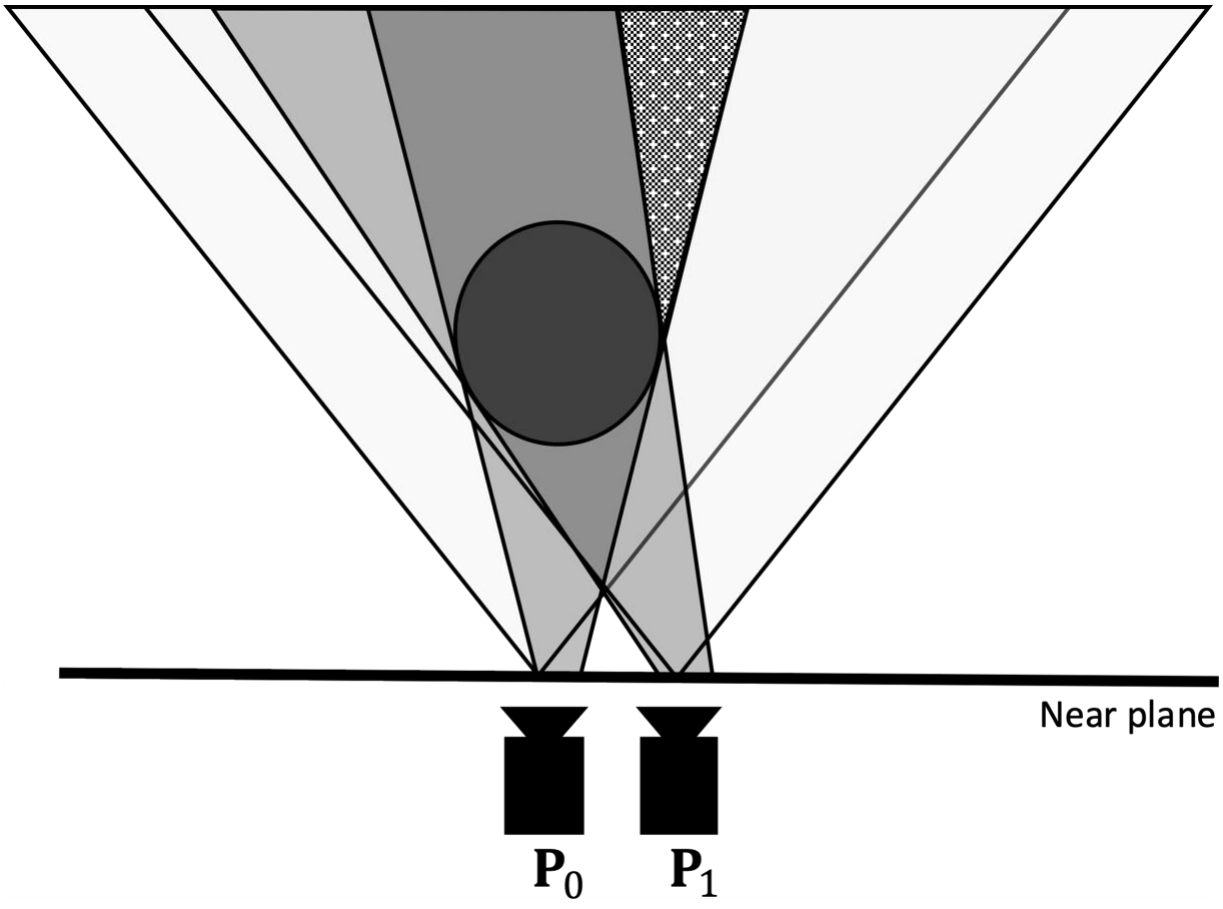


Рис.3.5 Область дизокклюзии между точками 0 и 1

Имея оба значения глубины, мы можем сравнить дельту между ними со значением разделения Экли (Akeley separation value). Интуитивно, константа разделения Экли обеспечивает минимальное расстояние между двумя объектами, представленными в буфере глубины с плавающей запятой, что позволяет с высокой степенью уверенности сказать, что объекты изначально отличались друг от друга.

На рисунке 3.6 видно, что средне-серые и темно-серые объекты имеют дельту, превышающую значение kSep, рассчитанное для конфигурации буфера глубины приложения. Однако расстояние от светло-серого объекта до средне-серого объекта не превышает вычисленного значения kSep, и поэтому мы не можем сделать вывод, отличается ли этот объект.

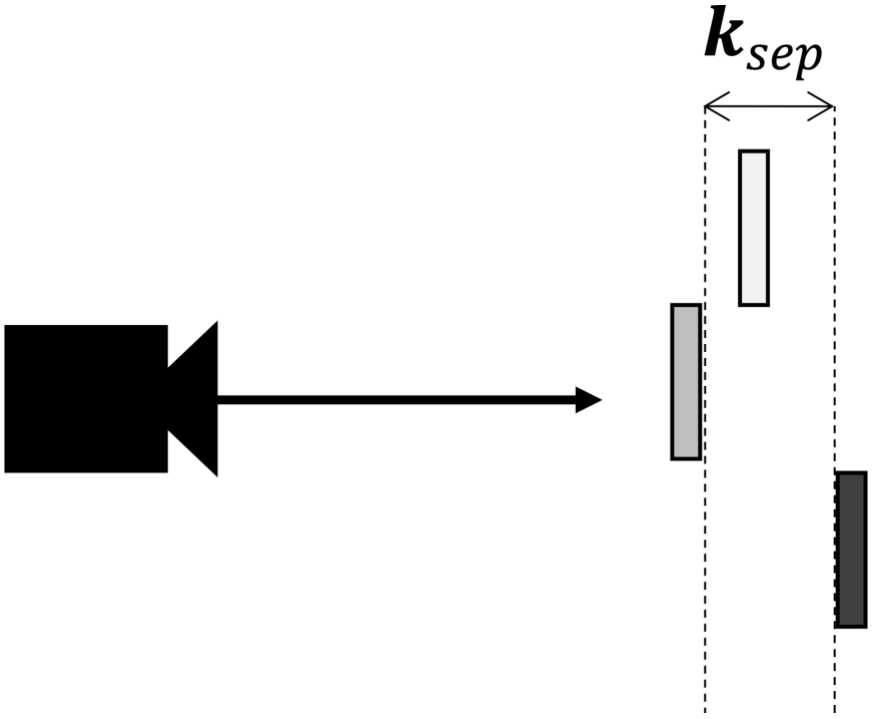


Рис.3.6 Концепт константы разделения

Значение, хранящееся в маске дизокклюзии, находится в диапазоне [0..1], где 1 соответствует значению, большему или равному значению разделения Экли.

3.2.5 Блокировка

Интуитивно понятно, что блокировка пикселя - это механизм, предотвращающий применение исправления цвета к пикселю. Чистый эффект этой блокировки заключается в том, что больше цветовых данных предыдущего кадра используется при вычислении окончательного цвета пикселя сверхвысокого разрешения на этапе перепроецирования и накопления. Текстура состояния блокировки содержит два значения, которые вместе составляют блокировку пикселя. Красный канал текстуры состояния блокировки содержит оставшееся время жизни блокировки пикселя. Это значение уменьшается на начальную длину блокировки, деленную на общую длину последовательности джиттера. Когда блокировка достигает нуля, она считается просроченной. Зеленый канал текстуры состояния блокировки содержит яркость пикселя во время создания блокировки, но он заполняется только на этапе повторного проецирования этапа перепроецирования и накопления. Значение яркости в конечном итоге используется на этапе перепроецирования и накопления как часть обнаружения изменения затенения, это позволяет FSR2 разблокировать пиксель, если во внешнем виде пикселя происходят прерывистые изменения.

3.2.6 Перепроецирование и накопление

Стадия перепроецирования и накопления (reproject and accumulate) FSR2 является наиболее сложной и дорогостоящей стадией в алгоритме. Она объединяет результаты многих предыдущих алгоритмических шагов и накапливает повторно спроецированные данные о цвете из предыдущего кадра вместе с данными о цвете с повышенной дискретизацией из текущего кадра. [4]

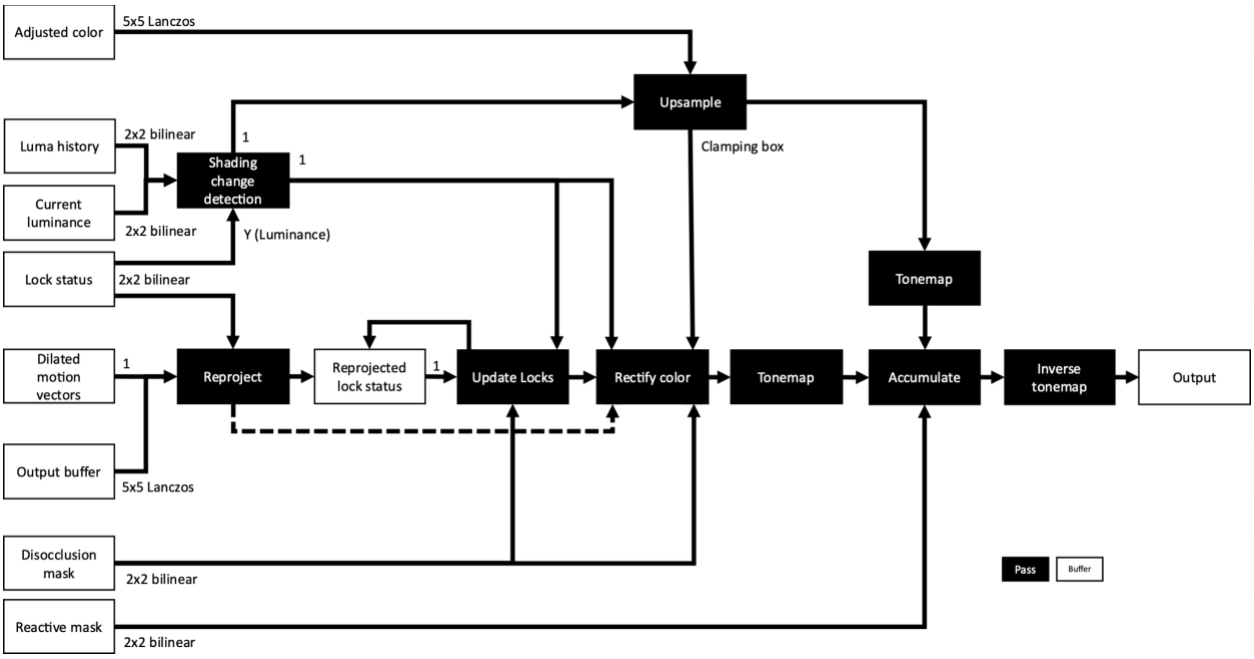


Рис.3.7 Алгоритм работы FSR

Первым шагом этапа перепроецирования и накопления является оценка каждого пикселя на наличие изменений в его затенении. Если мы находимся в заблокированной области, яркость во время создания блокировки сравнивается с порогом изменения затенения FSR2. В незаблокированной области для определения используются как текущий кадр, так и исторические значения яркости. Определение изменения затенения является ключевой частью этапа перепроецирования и накопления в FSR2 и используется во многих других частях этого этапа. [4]

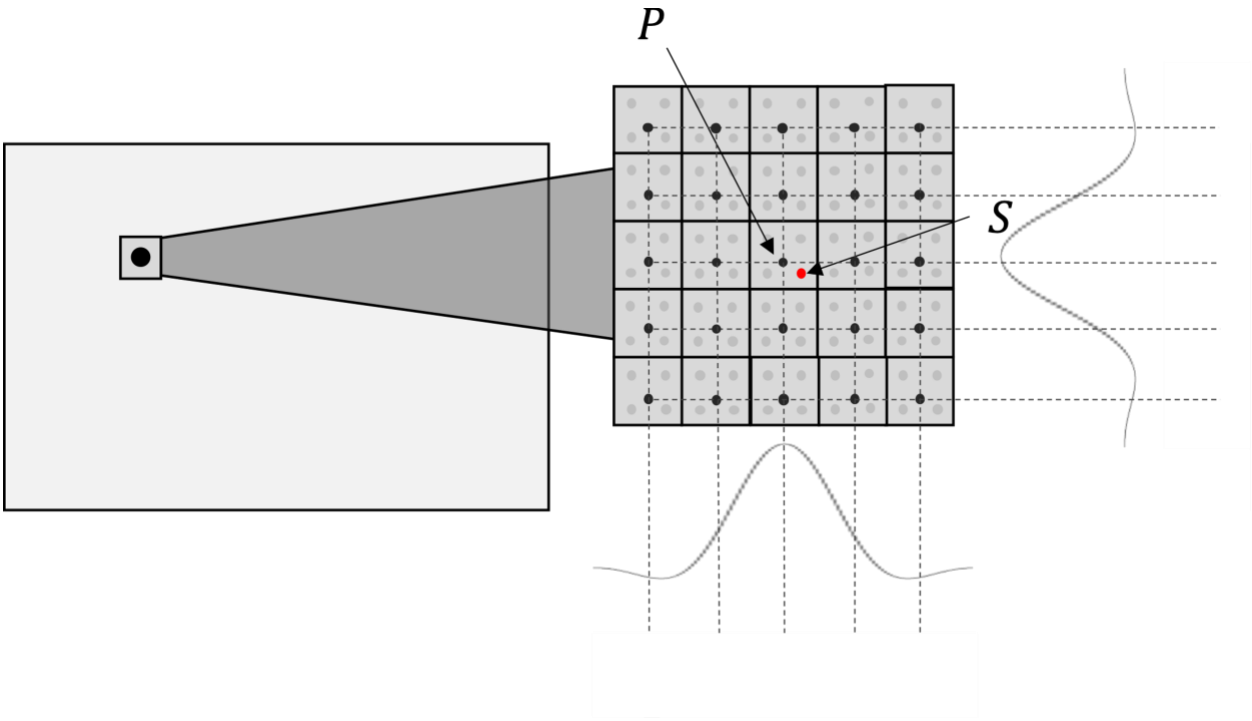


Рис.3.8 Апсемплинг с помощью выборки Ланцоша

Затем нужно увеличить частоту дискретизации скорректированного цвета. Для выполнения повышающей дискретизации положение пикселя скорректированного цвета служит центром ядра повторной выборки Ланцоша 5x5. На рисунке 3.8 можно видеть, что функции Ланцоша сосредоточены вокруг выборки разрешения дисплея S. Точка в каждом пикселе, помеченная P, обозначает положение выборки с дрожанием разрешения рендеринга, для которого мы вычисляем веса Ланцоша. Глядя выше и правее области 5x5 пикселей, можно увидеть, как ядро передискретизации Ланцоша(x, 2) применяется к образцам разрешения рендеринга в сетке пикселей 5x5, окружающих позицию пикселя. Поскольку шаг повышения дискретизации имеет доступ к окрестностям пикселей 5x5, с точки зрения эффективности имеет смысл также вычислить ограничивающую рамку YCoCg, которая используется во время коррекции цвета, на этом этапе. На рисунке 3.8 показана ограничивающая рамка 2D YCo, построенная из окрестности 3x3

вокруг текущего пикселя, в действительности ограничивающая рамка также имеет третье измерение для Cg. [4]

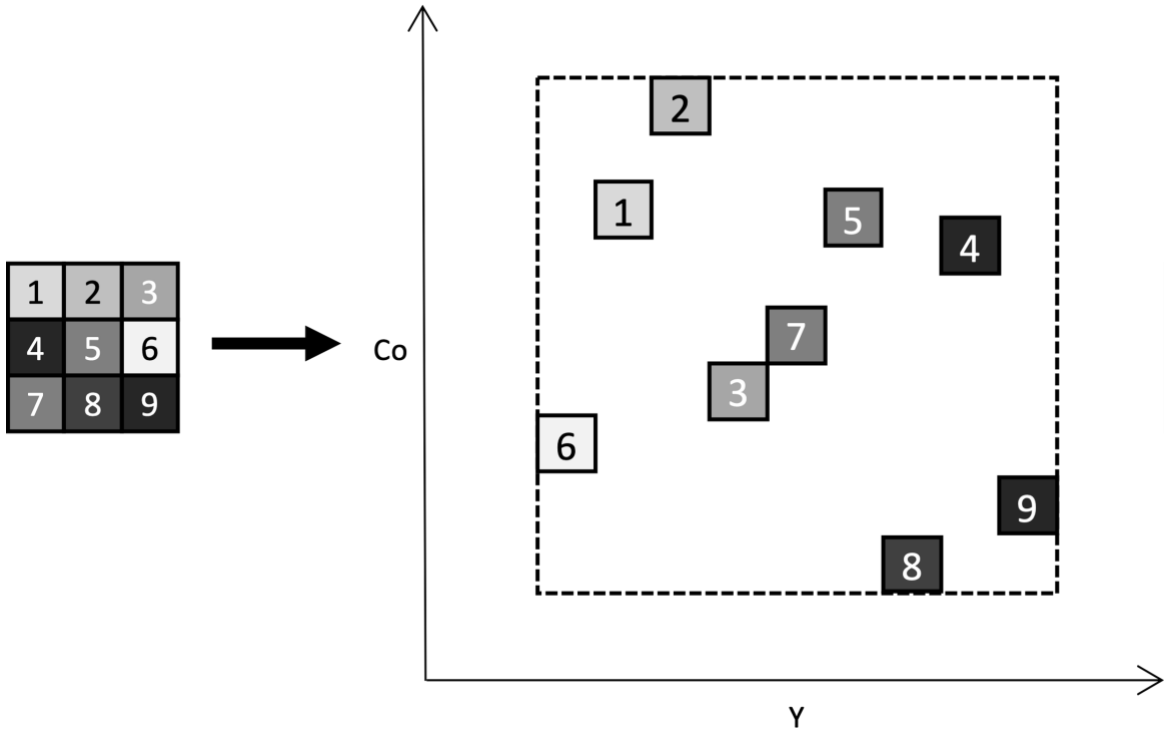


Рис.3.9 Ограничивающая рамка 2D YCo

Повторное проецирование — еще одна ключевая часть этапа перепроецирования и накопления. Чтобы выполнить повторное проецирование, расширенные векторы движения, созданные на этапе реконструкции и расширения, отбираются, а затем применяются к выходному буферу из предыдущего кадра, выполняющего FSR2.

Слева на рисунке 3.10 показан двумерный вектор движения M, примененный к текущей позиции пикселя. Результатом повторного проецирования является изображение с презентационным разрешением, которое содержит все данные из предыдущего кадра, которые могут быть отображены в текущем кадре. Однако перепроецируется не только выходной цвет предыдущего кадра. Поскольку FSR2 опирается на механизм, посредством которого каждый пиксель может быть заблокирован для повышения его временной

стабильности, блокировки также должны быть перепроецированы из предыдущего кадра в текущий кадр.

Это делается почти так же, как повторное проецирование данных о цвете, но также объединяет результаты шага обнаружения изменения затенения, который мы выполняли для различных значений яркости, как текущих, так и исторических. [4]

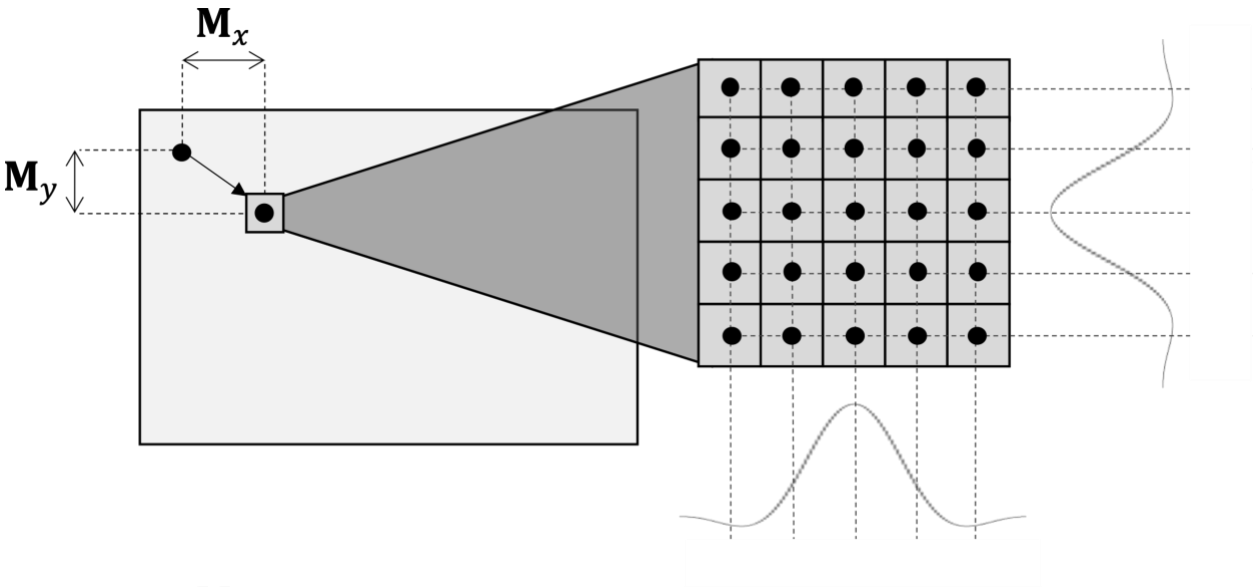


Рис.3.10 Вектор движения

Далее следует обновить блокировки. Первая задача обновления блокировок - поиск блокировок, созданных на этапе создания блокировок этого кадра, которые не перепроецируются, а вместо этого имеют значение яркости текущего кадра, записанное в зеленый канал перепроецированной текстуры замков. Все, что остается, - это определить, какие блокировки являются надежными для текущего кадра, и передать их на этап цветокоррекции. Определение достоверности выполняется путем сравнения значений яркости в окрестности пикселей в текущей текстуре яркости.

Применив обновления блокировки и определив их достоверность, можно перейти к исправлению цвета, что является следующим важным шагом этапа перепроецирования и накопления FSR2.

На этом этапе из исторических данных пикселя определяется окончательный цвет, который затем смешивается с цветом текущего кадра с повышением дискретизации, чтобы сформировать окончательный накопленный цвет сверхвысокого разрешения. Определение окончательного исторического цвета и его вклада в основном контролируются двумя вещами:

- Уменьшение влияния исторических образцов для областей, которые не окклюзированы. Это осуществляется путем модуляции значения цвета с помощью маски отключения.
- Уменьшение влияния исторических выборок (обозначенных Sh на рисунке 3.11) далеких от ограничивающей рамки текущего цвета кадра (вычисляется на этапе повышения частоты дискретизации на этапе перепроектирования и накопления). [4]

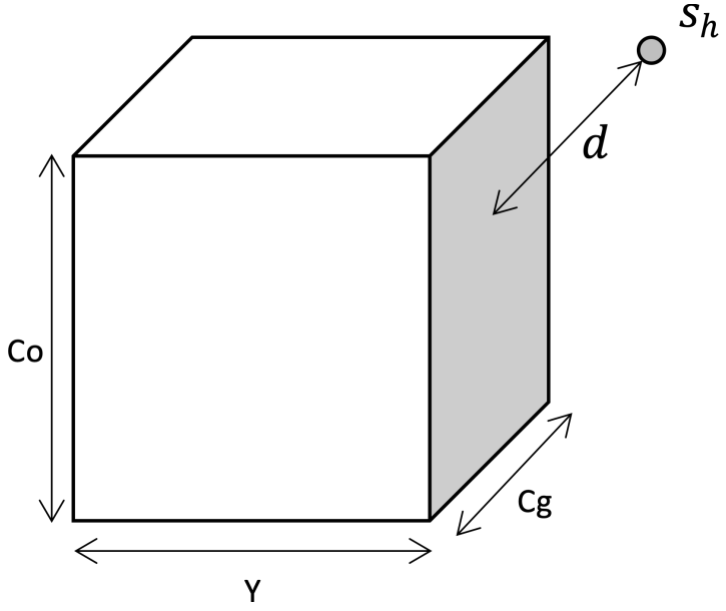


Рис.3.11 Сэмпл цвета зафиксированный в ограничивающий объем текущего кадра

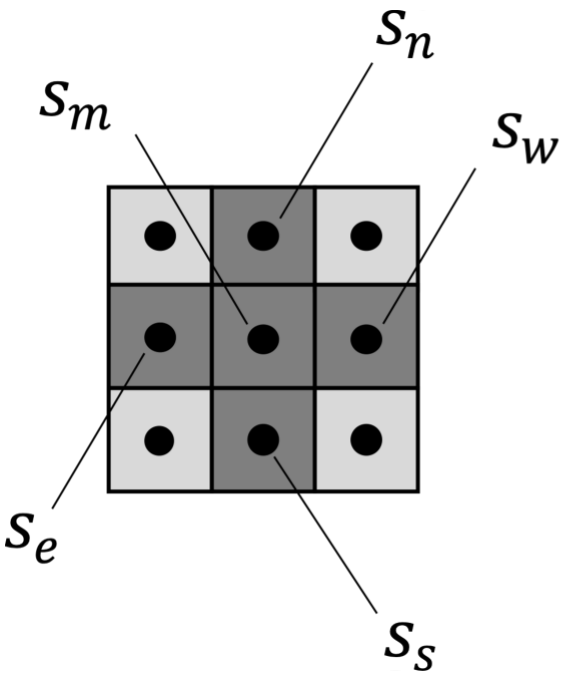
Последним этапом стадии перепроецирования и накопления является накопление цвета текущего кадра с повышением дискретизации с исправленными историческими данными о цвете. По умолчанию FSR2 обычно смешивает текущий кадр с относительно низким коэффициентом линейной интерполяции, т.е. относительно небольшая часть текущего кадра будет включена в конечный результат.

3.2.7 RCAS

Надежная контрастная адаптивная резкость (RCAS) изначально была представлена в FidelityFX Super Resolution 1.0 в качестве дополнительного прохода повышения резкости, помогающего повысить четкость и резкость конечного масштабированного изображения. RCAS является производным от популярного алгоритма Contrast Adaptive Sharpening (CAS), но с некоторыми ключевыми отличиями, которые делают его более подходящим для масштабирования.

В то время как CAS использует упрощенный механизм для преобразования локального контраста в переменную степень резкости, RCAS, наоборот, использует более точный механизм, определяя максимально возможную локальную резкость перед отсечением. Кроме того, RCAS также имеет встроенный процесс для ограничения резкости того, что он определяет как возможный шум. Поддержка некоторого масштабирования (которая была включена в CAS) не включена в RCAS, поэтому она должна работать с презентационным разрешением.

RCAS работает с данными, отобранными с помощью фильтра с 5 отводами, сконфигурированного по перекрестному шаблону.



$$w_0 = \frac{-s_m}{(s_n + s_e + s_s + s_w)}$$

$$w_1 = \frac{1 - s_m}{(s_n + s_e + s_s + s_w) - 4}$$

Рис.3.11 Веса RCAS примененные к соседним пикселям

После извлечения сэмплов RCAS затем выбирает «w», что приводит к отсутствию отсечения, ограничивает «w» и умножает на величину резкости. Приведенное выше решение имеет проблемы с вводом MSAA, поскольку шаги вдоль градиента вызывают проблемы с обнаружением краев. Чтобы помочь стабилизировать результаты RCAS, он использует 4-кратное максимальное и 4-кратное минимальное (в зависимости от уравнения) вместо отдельных касаний. [4]

5. Заключение

Алгоритмы DLSS и FSR отличаются принципом своей работы, но добиваются одной цели – улучшения графики. Решение подобной задачи очень важно для развития технологий улучшения видео и изображений на основе образца низкого разрешения.

Однако проблема улучшения изображения в реальном времени остается нерешенной до сих пор. Благодаря развитию нейросетей, решение этой задачи становится всё ближе.

6. Список использованной литературы

1. Wikipedia. Temporal anti-aliasing [Электронный ресурс]: свободная энциклопедия - /Wikipedia. - Электронные данные. Режим доступа: URL.: https://ru.wikipedia.org/wiki/Temporal_anti-aliasing, свободный - (дата обращения 10.11.2022)
2. Nvidia. NVIDIA DLSS 2.0: A Big Leap In AI Rendering [Электронный ресурс]: https://www.nvidia.com – Электронные данные. Режим доступа: URL.: https://www.nvidia.com/en-us/geforce/news/nvidia-dlss-2-0-a-big-leap-in-ai-rendering/, свободный - (дата обращения 10.11.2022)
3. Wikipedia. GPUOpen [Электронный ресурс]: свободная энциклопедия - /Wikipedia. - Электронные данные. Режим доступа: URL.: https://en.wikipedia.org/wiki/GPUOpen#FidelityFX, свободный - (дата обращения 10.11.2022)
4. GitHub. FidelityFX-FSR2 [Электронный ресурс]: https://github.com. - Электронные данные. Режим доступа: URL.: https://github.com/GPUOpen-Effects/FidelityFX-FSR2/blob/master/README.md, свободный - (дата обращения 10.11.2022)
5. Wikipedia. YCoCg [Электронный ресурс]: свободная энциклопедия - /Wikipedia. - Электронные данные. Режим доступа: URL.: https://en.wikipedia.org/wiki/YCoCg - (дата обращения 03.12.2022)
6. Appuals. Deep Learning Super Sampling (DLSS 2.0) Explained [Электронный ресурс]: https://appuals.com – Электронные данные. Режим доступа: URL.: https://appuals.com/deep-learning-super-sampling-dlss-2-0-explained/, свободный - (дата обращения 03.12.2022)
7. Behindthepixels. A Survey of Temporal Antialiasing Techniques [Электронный ресурс]: https:// behindthepixels.io – Электронные данные. Режим доступа: URL.: http://behindthepixels.io/assets/files/TemporalAA.pdf, свободный - (дата обращения 03.12.2022)