INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 4
MPEG VIDEO CODING

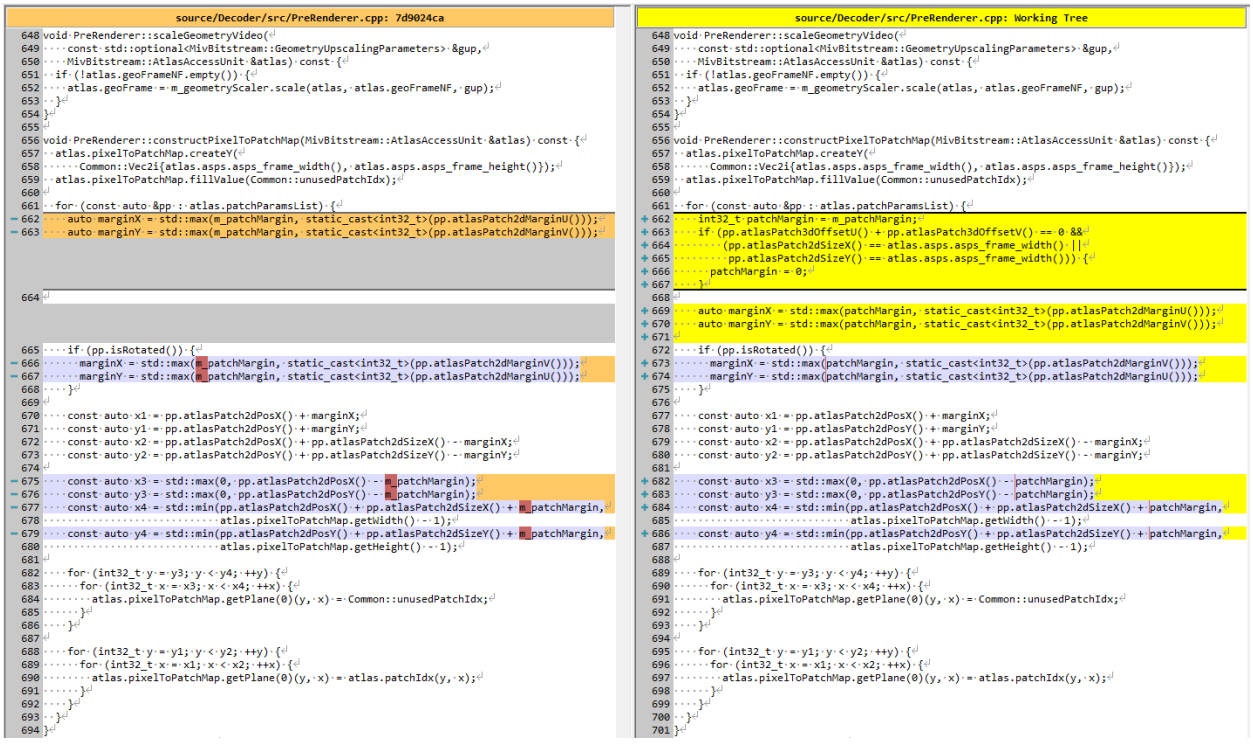
ISO/IEC JTC 1/SC 29/WG 4 m 70068
October 2024, Antalya, TR

Title: **[MIV] Selective patch margin enabling for rendering quality increase**
Source: **Adrian Dziembowski, Dominika Klóska, Błażej Szydełko, Dawid Mieloch (PUT), Gwangsoon Lee (ETRI)**

1 Abstract

The document presents an algorithm which allows for increasing the rendering quality by disabling the patch margin omitting tool for patches containing basic views. The algorithm significantly increases the objective and subjective quality of rendered basic views while having no negative impact on the quality of pose traces. The recommendation is to add the presented algorithm into the TMIV22.

2 Algorithm



source/Decoder/src/PreRenderer.cpp: 7d9024ca

```
void PreRenderer::scaleGeometryVideo(
    const std::optional<MivBitstream::GeometryUpscalingParameters> &gup,
    MivBitstream::AtlasAccessUnit &atlas) const {
  if (!atlas.geoFrameNF.empty()) {
    atlas.geoFrame = m_geometryScaler.scale(atlas, atlas.geoFrameNF, gup);
  }
}

void PreRenderer::constructPixelToPatchMap(MivBitstream::AtlasAccessUnit &atlas) const {
  atlas.pixelToPatchMap.createY(
      Common::Vec2i{atlas.asps.asps_frame_width(), atlas.asps.asps_frame_height()});
  atlas.pixelToPatchMap.fillValue(Common::unusedPatchIdx);

  for (const auto &pp : atlas.patchParamsList) {
-   auto marginX = std::max(m_patchMargin, static_cast<int32_t>(pp.atlasPatch2dMarginU()));
-   auto marginY = std::max(m_patchMargin, static_cast<int32_t>(pp.atlasPatch2dMarginV()));

    if (pp.isRotated()) {
-     marginX = std::max(m_patchMargin, static_cast<int32_t>(pp.atlasPatch2dMarginV()));
-     marginY = std::max(m_patchMargin, static_cast<int32_t>(pp.atlasPatch2dMarginU()));
    }

    const auto x1 = pp.atlasPatch2dPosX() + marginX;
    const auto y1 = pp.atlasPatch2dPosY() + marginY;
    const auto x2 = pp.atlasPatch2dPosX() + pp.atlasPatch2dSizeX() - marginX;
    const auto y2 = pp.atlasPatch2dPosY() + pp.atlasPatch2dSizeY() - marginY;

-   const auto x3 = std::max(0, pp.atlasPatch2dPosX() - m_patchMargin);
-   const auto y3 = std::max(0, pp.atlasPatch2dPosY() - m_patchMargin);
-   const auto x4 = std::min(pp.atlasPatch2dPosX() + pp.atlasPatch2dSizeX() + m_patchMargin,
                             atlas.pixelToPatchMap.getWidth() - 1);
-   const auto y4 = std::min(pp.atlasPatch2dPosY() + pp.atlasPatch2dSizeY() + m_patchMargin,
                             atlas.pixelToPatchMap.getHeight() - 1);

    for (int32_t y = y3; y < y4; ++y) {
      for (int32_t x = x3; x < x4; ++x) {
        atlas.pixelToPatchMap.getPlane(0)(y, x) = Common::unusedPatchIdx;
      }
    }

    for (int32_t y = y1; y < y2; ++y) {
      for (int32_t x = x1; x < x2; ++x) {
        atlas.pixelToPatchMap.getPlane(0)(y, x) = atlas.patchIdx(y, x);
      }
    }
  }
}
```

source/Decoder/src/PreRenderer.cpp: Working Tree

```
void PreRenderer::scaleGeometryVideo(
    const std::optional<MivBitstream::GeometryUpscalingParameters> &gup,
    MivBitstream::AtlasAccessUnit &atlas) const {
  if (!atlas.geoFrameNF.empty()) {
    atlas.geoFrame = m_geometryScaler.scale(atlas, atlas.geoFrameNF, gup);
  }
}

void PreRenderer::constructPixelToPatchMap(MivBitstream::AtlasAccessUnit &atlas) const {
  atlas.pixelToPatchMap.createY(
      Common::Vec2i{atlas.asps.asps_frame_width(), atlas.asps.asps_frame_height()});
  atlas.pixelToPatchMap.fillValue(Common::unusedPatchIdx);

  for (const auto &pp : atlas.patchParamsList) {
+   int32_t patchMargin = m_patchMargin;
+   if (pp.atlasPatch3dOffsetU() + pp.atlasPatch3dOffsetV() == 0 &&
+       (pp.atlasPatch2dSizeX() == atlas.asps.asps_frame_width() ||
+        pp.atlasPatch2dSizeY() == atlas.asps.asps_frame_width())) {
+     patchMargin = 0;
+   }

+   auto marginX = std::max(patchMargin, static_cast<int32_t>(pp.atlasPatch2dMarginU()));
+   auto marginY = std::max(patchMargin, static_cast<int32_t>(pp.atlasPatch2dMarginV()));

    if (pp.isRotated()) {
+     marginX = std::max(patchMargin, static_cast<int32_t>(pp.atlasPatch2dMarginV()));
+     marginY = std::max(patchMargin, static_cast<int32_t>(pp.atlasPatch2dMarginU()));
    }

    const auto x1 = pp.atlasPatch2dPosX() + marginX;
    const auto y1 = pp.atlasPatch2dPosY() + marginY;
    const auto x2 = pp.atlasPatch2dPosX() + pp.atlasPatch2dSizeX() - marginX;
    const auto y2 = pp.atlasPatch2dPosY() + pp.atlasPatch2dSizeY() - marginY;

+   const auto x3 = std::max(0, pp.atlasPatch2dPosX() - patchMargin);
+   const auto y3 = std::max(0, pp.atlasPatch2dPosY() - patchMargin);
+   const auto x4 = std::min(pp.atlasPatch2dPosX() + pp.atlasPatch2dSizeX() + patchMargin,
                             atlas.pixelToPatchMap.getWidth() - 1);
+   const auto y4 = std::min(pp.atlasPatch2dPosY() + pp.atlasPatch2dSizeY() + patchMargin,
                             atlas.pixelToPatchMap.getHeight() - 1);

    for (int32_t y = y3; y < y4; ++y) {
      for (int32_t x = x3; x < x4; ++x) {
        atlas.pixelToPatchMap.getPlane(0)(y, x) = Common::unusedPatchIdx;
      }
    }

    for (int32_t y = y1; y < y2; ++y) {
      for (int32_t x = x1; x < x2; ++x) {
        atlas.pixelToPatchMap.getPlane(0)(y, x) = atlas.patchIdx(y, x);
      }
    }
  }
}
```

3 Results

Mandatory content - Proposal vs. Low/High-bitrate Anchors

Sequence		BD-rate Y-PSNR	BD-rate IV-PSNR	BD-rate IV-SSIM	BD-PSNR Y-PSNR	BD-PSNR IV-PSNR	BD-SSIM IV-SSIM
Chess	B02	-5.1%	-5.3%	-0.1%	0.05	0.08	0.0000
Guitarist	B03	-4.7%	-3.3%	-0.3%	0.03	0.03	0.0001
Cadillac	J02	-25.2%	-30.3%	-2.0%	0.92	1.68	0.0003
Fan	J04	-10.4%	-10.0%	-0.4%	0.35	0.51	0.0001
Group	W01	-8.3%	-6.4%	-0.3%	0.13	0.22	0.0001
Painter	D01	-16.0%	-16.3%	-0.3%	0.70	0.87	0.0000
Frog	E01	-21.0%	-19.6%	-0.6%	0.72	1.04	0.0001
CBABasketball	L02	-14.7%	-15.0%	-0.7%	0.13	0.18	0.0000
Average		**-13.2%**	**-13.3%**	**-0.6%**	**0.38**	**0.58**	**0.0001**

Class A

Sequence		BD-rate Y-PSNR	BD-rate IV-PSNR	BD-rate IV-SSIM	BD-PSNR Y-PSNR	BD-PSNR IV-PSNR	BD-SSIM IV-SSIM
ClassroomVideo	A01	-6.8%	-2.8%	-0.2%	0.10	0.06	0.0000
Average		**-6.8%**	**-2.8%**	**-0.2%**	**0.10**	**0.06**	**0.0000**

Class B

Sequence		BD-rate Y-PSNR	BD-rate IV-PSNR	BD-rate IV-SSIM	BD-PSNR Y-PSNR	BD-PSNR IV-PSNR	BD-SSIM IV-SSIM
Museum	B01	-1.4%	-1.9%	-0.1%	0.03	0.08	0.0000
Chess	B02	-5.1%	-5.3%	-0.1%	0.05	0.08	0.0000
Guitarist	B03	-4.7%	-3.3%	-0.3%	0.03	0.03	0.0001
Average		**-3.7%**	**-3.5%**	**-0.2%**	**0.04**	**0.06**	**0.0000**

Class C

Sequence		BD-rate Y-PSNR	BD-rate IV-PSNR	BD-rate IV-SSIM	BD-PSNR Y-PSNR	BD-PSNR IV-PSNR	BD-SSIM IV-SSIM
Hijack	C01	-0.8%	-1.2%	-0.1%	0.02	0.02	0.0000
Cyberpunk	C02	-4.6%	-1.0%	0.9%	0.01	0.01	-0.0001
Average		**-2.7%**	**-1.1%**	**0.4%**	**0.01**	**0.02**	**-0.0001**

Class J

Sequence		BD-rate Y-PSNR	BD-rate IV-PSNR	BD-rate IV-SSIM	BD-PSNR Y-PSNR	BD-PSNR IV-PSNR	BD-SSIM IV-SSIM
Kitchen	J01	-37.8%	-46.7%	-1.4%	0.61	1.15	0.0001
Cadillac	J02	-25.2%	-30.3%	-2.0%	0.92	1.68	0.0003
Mirror	J03	-9.5%	-10.4%	-0.3%	0.36	0.60	0.0001
Fan	J04	-10.4%	-10.0%	-0.4%	0.35	0.51	0.0001
Average		**-20.7%**	**-24.3%**	**-1.0%**	**0.56**	**0.99**	**0.0001**

Class W

Sequence		BD-rate Y-PSNR	BD-rate IV-PSNR	BD-rate IV-SSIM	BD-PSNR Y-PSNR	BD-PSNR IV-PSNR	BD-SSIM IV-SSIM
Group	W01	-8.3%	-6.4%	-0.3%	0.13	0.22	0.0001
Dancing	W02	-13.9%	-15.9%	-1.2%	0.15	0.27	0.0001
Average		**-11.1%**	**-11.1%**	**-0.7%**	**0.14**	**0.25**	**0.0001**

Class D

Sequence		BD-rate Y-PSNR	BD-rate IV-PSNR	BD-rate IV-SSIM	BD-PSNR Y-PSNR	BD-PSNR IV-PSNR	BD-SSIM IV-SSIM
Painter	D01	-16.0%	-16.3%	-0.3%	0.70	0.87	0.0000
Breakfast	D02	-23.8%	-26.9%	-0.6%	0.44	0.70	0.0001
Barn	D03	-18.8%	-22.3%	-0.6%	0.25	0.39	0.0001
Average		**-19.6%**	**-21.8%**	**-0.5%**	**0.46**	**0.66**	**0.0001**

Class E

Sequence		BD-rate Y-PSNR	BD-rate IV-PSNR	BD-rate IV-SSIM	BD-PSNR Y-PSNR	BD-PSNR IV-PSNR	BD-SSIM IV-SSIM
Frog	E01	-21.0%	-19.6%	-0.6%	0.72	1.04	0.0001
Carpark	E02	-41.7%	-36.7%	-1.1%	2.58	3.01	0.0002
Street	E03	---	---	-1.8%	3.49	4.51	0.0001
Average		**---**	**---**	**-1.2%**	**2.26**	**2.85**	**0.0001**

Class L

Sequence		BD-rate Y-PSNR	BD-rate IV-PSNR	BD-rate IV-SSIM	BD-PSNR Y-PSNR	BD-PSNR IV-PSNR	BD-SSIM IV-SSIM
Fencing	L01	-6.2%	-8.0%	-0.2%	0.18	0.26	0.0000
CBABasketball	L02	-14.7%	-15.0%	-0.7%	0.13	0.18	0.0000
MartialArts	L03	-21.1%	-29.1%	-0.9%	0.19	0.29	0.0001
Average		**-14.0%**	**-17.3%**	**-0.6%**	**0.17**	**0.24**	**0.0000**

Anchor:

Proposal:

Difference between anchor and proposal, J01:

4 Recommendation

The recommendation is to add the presented algorithm into the TMIV22.

5 Acknowledgement

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2018-0-00207, Immersive Media Research Laboratory).