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 m57563
July 2021, Online

Title: **Corrected quaternion to Euler angle conversion for MIV DSDE**

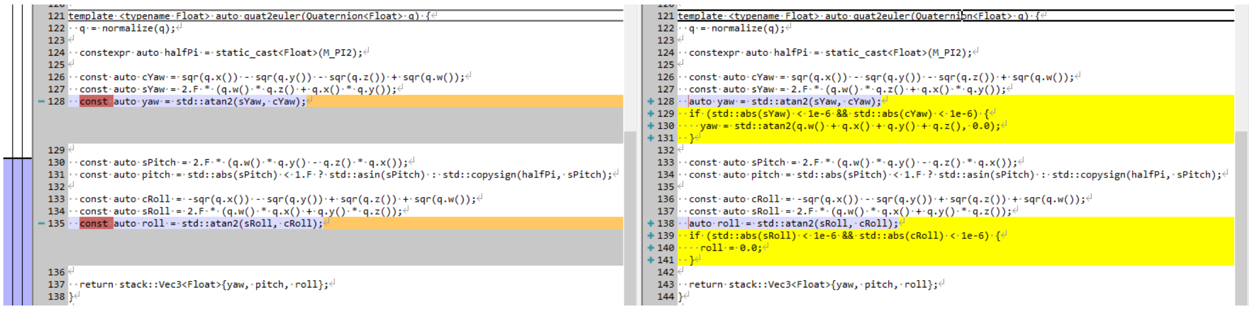
Source: **Adrian Dziembowski, Dawid Mieloch (Poznań University of Technology)**

Abstract

The document presents the description of the fix of TMIV9.1, which allows to properly convert quaternions to Euler angles for cameras facing up and down in sequences SN and SQ thus to output proper camera parameters at the decoder side using the MIV DSDE profile. The fix impacts only the G17 anchor, sequences SN and SQ. It does not have any impact on other content.

1 The fix

Only *Quaternion.h* and *Quaternion.test.cpp* were changed. Left – TMIV9.1, right – fix.



```
template <typename Float> auto quat2euler(Quaternion<Float> q) {
  q = normalize(q);

  constexpr auto halfPi = static_cast<Float>(M_PI2);

  const auto cYaw = sqr(q.x()) - sqr(q.y()) - sqr(q.z()) + sqr(q.w());
  const auto sYaw = 2.F * (q.w() * q.z() + q.x() * q.y());
  const auto yaw = std::atan2(sYaw, cYaw);

  const auto sPitch = 2.F * (q.w() * q.y() - q.z() * q.x());
  const auto pitch = std::abs(sPitch) < 1.F ? std::asin(sPitch) : std::copysign(halfPi, sPitch);

  const auto cRoll = -sqr(q.x()) - sqr(q.y()) + sqr(q.z()) + sqr(q.w());
  const auto sRoll = 2.F * (q.w() * q.x() + q.y() * q.z());
  const auto roll = std::atan2(sRoll, cRoll);

  return stack::Vec3<Float>{yaw, pitch, roll};
}
```

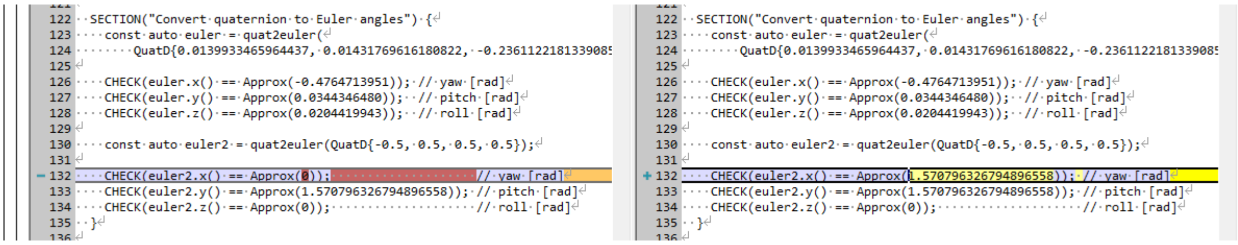
```
template <typename Float> auto quat2euler(Quaternion<Float> q) {
  q = normalize(q);

  constexpr auto halfPi = static_cast<Float>(M_PI2);

  const auto cYaw = sqr(q.x()) - sqr(q.y()) - sqr(q.z()) + sqr(q.w());
  const auto sYaw = 2.F * (q.w() * q.z() + q.x() * q.y());
  auto yaw = std::atan2(sYaw, cYaw);
  if (std::abs(sYaw) < 1e-6 && std::abs(cYaw) < 1e-6) {
    yaw = std::atan2(q.w() + q.x() + q.y() + q.z(), 0.0);
  }

  const auto sPitch = 2.F * (q.w() * q.y() - q.z() * q.x());
  const auto pitch = std::abs(sPitch) < 1.F ? std::asin(sPitch) : std::copysign(halfPi, sPitch);

  const auto cRoll = -sqr(q.x()) - sqr(q.y()) + sqr(q.z()) + sqr(q.w());
  const auto sRoll = 2.F * (q.w() * q.x() + q.y() * q.z());
  auto roll = std::atan2(sRoll, cRoll);
  if (std::abs(sRoll) < 1e-6 && std::abs(cRoll) < 1e-6) {
    roll = 0.0;
  }

  return stack::Vec3<Float>{yaw, pitch, roll};
}
```

In TMIV9.1 the yaw and roll could be (and were, for 2 views of SN and SQ) incorrectly estimated because of calculation of atan2(0, 0), which is undefined in its nature.

The yaw is calculated as atan2(sYaw, cYaw), the roll – as atan2(sRoll, cRoll).

For view facing up (v0), the quaternion is: [-0.5, -0.5, -0.5, 0.5]. For v9 (facing down), the quaternion is: [-0.5, 0.5, 0.5, 0.5]. In both cases, sYaw, cYaw, sRoll, and cRoll were equal to (or close to) zero.

The fix checks, if the view is facing along the vertical axis by checking the values of sYaw, cYaw, sRoll, and cRoll. If they are 0, the value of yaw and roll are updated.

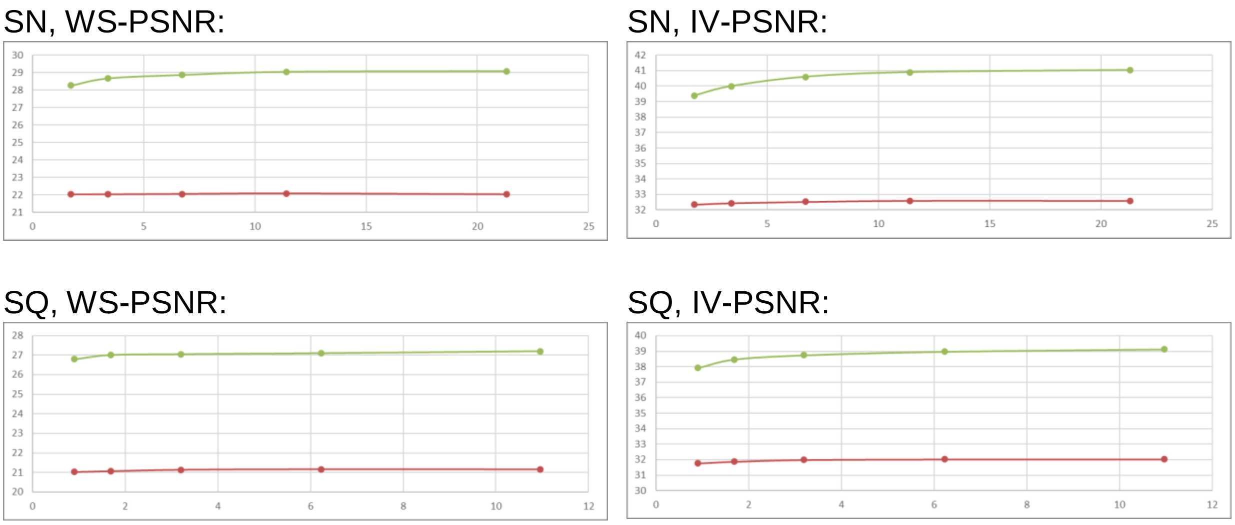


```
  SECTION("Convert quaternion to Euler angles") {
    const auto euler = quat2euler(
        QuatD{0.0139933465964437, 0.0143176961618082, -0.236112218133908

    CHECK(euler.x() == Approx(-0.4764713951)); // yaw [rad]
    CHECK(euler.y() == Approx(0.0344346480));  // pitch [rad]
    CHECK(euler.z() == Approx(0.0204419943));  // roll [rad]

    const auto euler2 = quat2euler(QuatD{-0.5, 0.5, 0.5, 0.5});

    CHECK(euler2.x() == Approx(0));                    // yaw [rad]
    CHECK(euler2.y() == Approx(1.570796326794896558)); // pitch [rad]
    CHECK(euler2.z() == Approx(0));                    // roll [rad]
  }
```

```
  SECTION("Convert quaternion to Euler angles") {
    const auto euler = quat2euler(
        QuatD{0.0139933465964437, 0.0143176961618082, -0.236112218133908

    CHECK(euler.x() == Approx(-0.4764713951)); // yaw [rad]
    CHECK(euler.y() == Approx(0.0344346480));  // pitch [rad]
    CHECK(euler.z() == Approx(0.0204419943));  // roll [rad]

    const auto euler2 = quat2euler(QuatD{-0.5, 0.5, 0.5, 0.5});

    CHECK(euler2.x() == Approx(1.570796326794896558)); // yaw [rad]
    CHECK(euler2.y() == Approx(1.570796326794896558)); // pitch [rad]
    CHECK(euler2.z() == Approx(0));                    // roll [rad]
  }
```

In the unit test, the proper 90 degrees are now set instead of 0. “Proper” – the same rotation, as in input camera parameters.

2 Results



SN, WS-PSNR:

SN, IV-PSNR:

SQ, WS-PSNR:

SQ, IV-PSNR:

Fig. 1. RD-curves for SN and SQ sequences, G17; red: TMIV9.1, green: proposed fix.

The proposed fix was tested under G17 configuration for all content.

Mandatory content - Proposal vs. Low/High-bitrate Anchors										Runtime ratio (%)		
Sequence		High-BR BD rate Y-PSNR	Low-BR BD rate Y-PSNR	Max delta Y-PSNR	High-BR BD rate IV-PSNR	Low-BR BD rate IV-PSNR	Pixel rate [%]	Pixel rate [GP/s]	Frame rate [Hz]	Atlas encoding	Video encoding	Decoding & Rendering
ClassroomVideo	A	0.0%	0.0%	5.66	0.0%	0.0%	100%	1.07	30	95.4%	81.1%	85.7%
Museum	B	0.0%	0.0%	9.70	0.0%	0.0%	100%	1.07	30	96.2%	88.2%	92.5%
Fan	O	0.0%	0.0%	10.81	0.0%	0.0%	100%	1.07	30	71.8%	73.2%	85.5%
Kitchen	J	0.0%	0.0%	11.74	0.0%	0.0%	100%	1.07	30	87.4%	86.4%	94.3%
Painter	D	0.0%	0.0%	8.99	0.0%	0.0%	100%	1.07	30	87.3%	71.9%	88.3%
Frog	E	0.0%	0.0%	7.61	0.0%	0.0%	100%	1.07	30	99.0%	99.0%	44.5%
Carpark	P	0.0%	0.0%	11.01	0.0%	0.0%	83%	0.89	25	81.5%	89.3%	70.3%
Chess	N	---	---	24.11	---	---	100%	1.07	30	73.3%	90.0%	86.9%
Group	R	0.0%	0.0%	22.62	0.0%	0.0%	100%	1.07	30	77.6%	80.3%	93.2%
MIV		---	---	**12.47**	---	---	98%	1.05		85.5%	84.4%	82.3%

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

Fencing	L	0.0%	0.0%	13.35	0.0%	0.0%	83%	0.89	25	83.4%	65.0%	72.2%
Hall	T	0.0%	0.0%	18.55	0.0%	0.0%	83%	0.89	25	71.0%	70.2%	101.7%
Street	U	0.0%	0.0%	7.02	0.0%	0.0%	83%	0.89	25	78.4%	72.9%	63.0%
ChessPieces	Q	---	---	27.84	---	---	100%	1.07	30	85.4%	88.6%	86.5%
Hijack	C	0.0%	0.0%	21.92	0.0%	0.0%	100%	1.07	30	97.3%	71.8%	79.9%
Mirror	I	0.0%	0.0%	13.51	0.0%	0.0%	100%	1.07	30	73.3%	72.7%	74.0%
MIV		---	---	**17.03**	---	---	92%	0.98		81.5%	73.5%	79.5%

3 Recommendation

We recommend to include the proposed fix into TMIV.

Acknowledgement

The research was supported by the Ministry of Science and Higher Education of Republic of Poland.