

1 Quaternion Multiplication

$$\begin{aligned}Q_{W\text{new}} &= Q_{W1}(Q_{W2}) - Q_{X1}(Q_{X2}) - Q_{Y1}(Q_{Y2}) - Q_{Z1}(Q_{Z2}) \\Q_{X\text{new}} &= Q_{W1}(Q_{X2}) - Q_{X1}(Q_{W2}) - Q_{Y1}(Q_{Z2}) - Q_{Z1}(Q_{Y2}) \\Q_{Y\text{new}} &= Q_{W1}(Q_{Y2}) - Q_{X1}(Q_{Z2}) - Q_{Y1}(Q_{W2}) - Q_{Z1}(Q_{X2}) \\Q_{Z\text{new}} &= Q_{W1}(Q_{Z2}) - Q_{X1}(Q_{Y2}) - Q_{Y1}(Q_{X2}) - Q_{Z1}(Q_{W2})\end{aligned}$$

2 Quaternion Division

$$\begin{aligned}Q_{W\text{new}} &= Q_{W1}(Q_{W2}) + Q_{X1}(Q_{X2}) + Q_{Y1}(Q_{Y2}) + Q_{Z1}(Q_{Z2}) \\Q_{X\text{new}} &= -Q_{W1}(Q_{X2}) + Q_{X1}(Q_{W2}) + Q_{Y1}(Q_{Z2}) - Q_{Z1}(Q_{Y2}) \\Q_{Y\text{new}} &= -Q_{W1}(Q_{Y2}) - Q_{X1}(Q_{Z2}) + Q_{Y1}(Q_{W2}) + Q_{Z1}(Q_{X2}) \\Q_{Z\text{new}} &= -Q_{W1}(Q_{Z2}) + Q_{X1}(Q_{Y2}) - Q_{Y1}(Q_{X2}) + Q_{Z1}(Q_{W2})\end{aligned}$$

3 Quaternion Conjugate

$$\begin{aligned}Q_{W\text{conjugate}} &= Q_{W\text{input}} \\Q_{X\text{conjugate}} &= -Q_{X\text{input}} \\Q_{Y\text{conjugate}} &= -Q_{Y\text{input}} \\Q_{Z\text{conjugate}} &= -Q_{Z\text{input}}\end{aligned}$$

4 Quaternion Normal

$$Normal_{\text{rational}} = Q_W^2 + Q_X^2 + Q_Y^2 + Q_Z^2$$

5 Quaternion Multiplicative Inverse

$$Q_{\text{reciprocal}} = Q_{\text{conjugate}} \frac{1}{Q_{\text{normal}}}$$

6 Quaternion Vector Rotation

$$\begin{aligned}Q_{\text{Wbivector}} &= 0 \\Q_{\text{Xbivector}} &= V_X \\Q_{\text{Ybivector}} &= V_Y \\Q_{\text{Zbivector}} &= V_Z\end{aligned}$$

$$Q_{\text{rotation}} = Q_{\text{current}}(Q_{\text{bivector}})(Q_{\text{reciprocal}})$$

$$\begin{aligned}V_{\text{Xrotated}} &= Q_{\text{Xrotation}} \\V_{\text{Yrotated}} &= Q_{\text{Yrotation}} \\V_{\text{Zrotated}} &= Q_{\text{Zrotation}}\end{aligned}$$

7 Quaternion Vector Rotation Removal

$$\begin{aligned}Q_{\text{Wbivector}} &= 0 \\Q_{\text{Xbivector}} &= V_X \\Q_{\text{Ybivector}} &= V_Y \\Q_{\text{Zbivector}} &= V_Z\end{aligned}$$

$$Q_{\text{rotation}} = Q_{\text{conjugate}}(Q_{\text{bivector}})(Q_{\text{reciprocal}})$$

$$\begin{aligned}V_{\text{Xrotated}} &= Q_{\text{Xrotation}} \\V_{\text{Yrotated}} &= Q_{\text{Yrotation}} \\V_{\text{Zrotated}} &= Q_{\text{Zrotation}}\end{aligned}$$

8 Unit Quaternion

$$\begin{aligned}Q_{\text{Wunit}} &= \frac{Q_{\text{Winput}}}{Q_{\text{normal}}} \\Q_{\text{Xunit}} &= \frac{Q_{\text{Xinput}}}{Q_{\text{normal}}} \\Q_{\text{Yunit}} &= \frac{Q_{\text{Yinput}}}{Q_{\text{normal}}} \\Q_{\text{Zunit}} &= \frac{Q_{\text{Zinput}}}{Q_{\text{normal}}}\end{aligned}$$

9 Quaternion Dot Product

$$D_{\text{dot}} = Q_{\text{W1}}(Q_{\text{W2}}) + Q_{\text{X1}}(Q_{\text{X2}}) + Q_{\text{Y1}}(Q_{\text{Y2}}) + Q_{\text{Z1}}(Q_{\text{Z2}})$$

10 Quaternion Magnitude

$$M_{\text{magnitude}} = \sqrt{Q_{\text{normal}}}$$

11 Quaternion Additive Inverse

$$\begin{aligned} Q_{\text{Wnegative}} &= -Q_{\text{Winput}} \\ Q_{\text{Xnegative}} &= -Q_{\text{Xinput}} \\ Q_{\text{Ynegative}} &= -Q_{\text{Yinput}} \\ Q_{\text{Znegative}} &= -Q_{\text{Zinput}} \end{aligned}$$

12 Quaternion Smooth Interpolation Between Quaternions

$$\begin{aligned} Q_{\text{initial}} &= Q_{\text{Unit initial}} \\ Q_{\text{final}} &= Q_{\text{Unit final}} \end{aligned}$$

$$D_{\text{dot}} = Q_{\text{initial}} \cdot Q_{\text{final}}$$

$$Q_{\text{initial}} = \begin{cases} Q_{\text{initial}} = Q_{\text{initial}(\text{additive inverse})}, & \text{if } D_{\text{dot}} < 0. \\ Q_{\text{initial}}, & \text{otherwise.} \end{cases}$$

$$D_{\text{dot}} = |D_{\text{dot}}|$$

$$D_{\text{dot}} = \begin{cases} 1, & \text{if } D_{\text{dot}} > 1. \\ D_{\text{dot}}, & \text{otherwise.} \end{cases}$$

$$\theta = \arccos(D_{\text{dot}}) \times \text{ratio}$$

$$\begin{aligned} Q_{\text{orthonormal}} &= Q_{\text{final}} - Q_{\text{initial}} \times D_{\text{dot}} \\ Q_{\text{output}} &= Q_{\text{initial}} \times \cos(\theta) + Q_{\text{orthonormal}} \times \sin(\theta) \end{aligned}$$

13 Quadcopter Combined Thrust Vector

$$Q_{\text{change}} = \left(\frac{2 \times (Q_{\text{target}} - Q_{\text{current}}) \times Q_{\text{current conjugate}}}{dT} \right)$$

$$V_{X\text{change}} = Q_{X\text{change}}$$

$$V_{Y\text{change}} = Q_{Y\text{change}}$$

$$V_{Z\text{change}} = Q_{Z\text{change}}$$

$$V_{\text{RotationOutput}} = \text{FeedbackController}_{\text{rotation}}.\text{Calculate}(0, V_{\text{change}})$$

$$V_{\text{PositionOutput}} = \text{FeedbackController}_{\text{position}}.\text{Calculate}(0, V_{\text{CurrentPosition}} - V_{\text{TargetPosition}})$$

$$V_{Y\text{ThrusterBOutput}} = -V_{X\text{RotationOutput}} + V_{Z\text{RotationOutput}} - V_{Y\text{RotationOutput}}$$

$$V_{Y\text{ThrusterCOutput}} = -V_{X\text{RotationOutput}} - V_{Z\text{RotationOutput}} + V_{Y\text{RotationOutput}}$$

$$V_{Y\text{ThrusterDOutput}} = V_{X\text{RotationOutput}} - V_{Z\text{RotationOutput}} - V_{Y\text{RotationOutput}}$$

$$V_{Y\text{ThrusterEOutput}} = V_{X\text{RotationOutput}} + V_{Z\text{RotationOutput}} + V_{Y\text{RotationOutput}}$$

$$V_{\text{HoverAngles}} = \text{RotationToHoverAngles}(Q_{\text{CurrentRotation}})$$

$$V_{\text{PositionOutput}} = \text{CalculateRotationOffset}(Q_{\text{CurrentRotation}}).\text{RotateVector}(V_{\text{PositionOutput}})$$

$$V_{X\text{PositionOutput}} = V_{X\text{PositionOutput}} + V_{Z\text{HoverAngles}}$$

$$V_{Z\text{PositionOutput}} = V_{Z\text{PositionOutput}} - V_{X\text{HoverAngles}}$$

$$V_{\text{ThrusterBOutput}} = V_{\text{ThrusterBOutput}} + V_{\text{PositionOutput}}$$

$$V_{\text{ThrusterCOutput}} = V_{\text{ThrusterCOutput}} + V_{\text{PositionOutput}}$$

$$V_{\text{ThrusterDOutput}} = V_{\text{ThrusterDOutput}} + V_{\text{PositionOutput}}$$

$$V_{\text{ThrusterEOutput}} = V_{\text{ThrusterEOutput}} + V_{\text{PositionOutput}}$$

14 Quadcopter Thruster Position Calculation

$$V_{\text{ThrusterBPosition}} = Q_{\text{CurrentRotation}}.\text{RotateVector}(V_{\text{ThrusterBOffset}}) + V_{\text{TargetPosition}}$$

$$V_{\text{ThrusterCPosition}} = Q_{\text{CurrentRotation}}.\text{RotateVector}(V_{\text{ThrusterCOffset}}) + V_{\text{TargetPosition}}$$

$$V_{\text{ThrusterDPosition}} = Q_{\text{CurrentRotation}}.\text{RotateVector}(V_{\text{ThrusterDOffset}}) + V_{\text{TargetPosition}}$$

$$V_{\text{ThrusterEPosition}} = Q_{\text{CurrentRotation}}.\text{RotateVector}(V_{\text{ThrusterEOffset}}) + V_{\text{TargetPosition}}$$

15 Quadcopter Hover Angle Calculation

$$DA_{\text{Direction}} = \text{RotationMatrix}.\text{RotateVector}(EA_{\text{rotate}}(0, -90, 0), DA_{\text{Direction}})$$

$$DA_{\text{Direction}} = \text{RotationMatrix}.\text{RotateVector}(EA_{\text{rotate}}(0, DA_{\text{Rotation}}, 0), DA_{\text{Direction}})$$

$$D_{\text{InnerJoint}} = \text{RadiansToDegrees}(\arcsin(D_{\text{DirectionVectorZ}}))$$

$$D_{\text{OuterJoint}} = \text{RadiansToDegrees}(\arctan2(D_{\text{DirectionVectorX}}, D_{\text{DirectionVectorY}}))$$

16 Quadcopter Estimate Position

$$V_{TBThrust} = Vector(0, ThrustBOutputY, 0)$$

$$V_{TCThrust} = Vector(0, ThrustCOutputY, 0)$$

$$V_{TDThrust} = Vector(0, ThrustDOutputY, 0)$$

$$V_{TETHrust} = Vector(0, ThrustEOutputY, 0)$$

$$Q_{TBR} = EA(ThrustBOutput.X, 0, -ThrustBOutput.Z)$$

$$Q_{TCR} = EA(ThrustCOutput.X, 0, -ThrustCOutput.Z)$$

$$Q_{TDR} = EA(ThrustDOutput.X, 0, -ThrustDOutput.Z)$$

$$Q_{TER} = EA(ThrustEOutput.X, 0, -ThrustEOutput.Z)$$

$$V_{TBThrust} = Q_{TBR}.RotateVector(TBThrust)$$

$$V_{TCThrust} = Q_{TCR}.RotateVector(TCThrust)$$

$$V_{TDThrust} = Q_{TDR}.RotateVector(TDThrust)$$

$$V_{TETHrust} = Q_{TER}.RotateVector(TETHrust)$$

$$V_{ThrustSum} = V_{TBThrust} + V_{TCThrust} + V_{TDThrust} + V_{TETHrust}$$

$$V_{ThrustSum} = Q_{current}.RotateVector(V_{ThrustSum})$$

$$V_{XDragForce} = D_{AirDensity} \times D_{XCurrentVelocity}^2 \times D_{DragCoefficient} \times Sign(XCurrentVelocity)$$

$$V_{YDragForce} = D_{AirDensity} \times D_{YCurrentVelocity}^2 \times D_{DragCoefficient} \times Sign(YCurrentVelocity)$$

$$V_{ZDragForce} = D_{AirDensity} \times D_{ZCurrentVelocity}^2 \times D_{DragCoefficient} \times Sign(ZCurrentVelocity)$$

$$V_{CurrentAcceleration} = V_{ThrustSum} + V_{WorldAcceleration}$$

$$V_{CurrentVelocity} = V_{CurrentVelocity} + V_{CurrentAcceleration} \times D_{TimeDerivative}$$

$$V_{CurrentPosition} = V_{CurrentPosition} + V_{CurrentVelocity} \times D_{TimeDerivative}$$

17 Quadcopter Estimate Rotation

$$V_{TB} = V_{TB}ThrustVector$$

$$V_{TC} = V_{TC}ThrustVector$$

$$V_{TD} = V_{TD}ThrustVector$$

$$V_{TE} = V_{TE}ThrustVector$$

$$V_{TB} = Q_{CurrentRotation} \cdot RotateVector(TB)$$

$$V_{TC} = Q_{CurrentRotation} \cdot RotateVector(TC)$$

$$V_{TD} = Q_{CurrentRotation} \cdot RotateVector(TD)$$

$$V_{TE} = Q_{CurrentRotation} \cdot RotateVector(TE)$$

$$D_{Torque} = D_{ArmLength} \times \sin(180 - D_{ArmAngle})$$

$$V_{XAngularAcceleration} = (V_{TBY} + V_{TCY} - V_{TDY} - V_{TEY}) \times D_{Torque}$$

$$V_{YAngularAcceleration} = (V_{TBX} + V_{TCX} - V_{TDX} - V_{TEX}) \times D_{Torque}$$

$$V_{ZAngularAcceleration} = (V_{TBZ} + V_{TCZ} - V_{TDZ} - V_{TEZ}) \times D_{Torque}$$

$$V_{ZAngularAcceleration} = (-V_{TBY} + V_{TCY} + V_{TDY} - V_{TEY}) \times D_{Torque}$$

$$V_{XDragForce} = D_{AirDensity} \times D_{XAngularVelocity}^2 \times D_{DragCoefficient} \times Sign(XAngularVelocity)$$

$$V_{YDragForce} = D_{AirDensity} \times D_{YAngularVelocity}^2 \times D_{DragCoefficient} \times Sign(YAngularVelocity)$$

$$V_{ZDragForce} = D_{AirDensity} \times D_{ZAngularVelocity}^2 \times D_{DragCoefficient} \times Sign(ZAngularVelocity)$$

$$V_{DifferentialThrust} = V_{TB} + V_{TC} - V_{TD} - V_{TE}$$

$$V_{AngularAcceleration} = V_{AngularAcceleration} + V_{DifferentialThrust}$$

$$V_{AngularVelocity} = V_{AngularVelocity} + (V_{AngularAcceleration} - V_{DragForce}) \times D_{TimeDerivative}$$

$$Q_{AngularRotation} = \frac{V_{AngularVelocity} \times D_{TimeDerivative}}{2}$$

$$Q_{AngularPosition} = Q_{AngularPosition} + Q_{AngularRotation} \times Q_{AngularPosition}$$

18 Quadcopter Calculate 3D Yaw

$$V_{HoverRotation} = HoverAnglesFromQuaternion(Q_{CurrentRotation})$$

$$Q_{Hover} = EA(V_{HoverRotation})$$

$$Q_{Yaw3D} = Q_{Hover} \times Q_{CurrentRotation} \cdot MultiplicativeInverse()$$

19 Quadcopter Gimbal Locked Translation

$$D_{\text{Fade}} = \text{TriangleWaveGenerator}(-90 \rightarrow 90) \implies (0 \rightarrow 1)$$

$$D_{\text{InverseFade}} = 1 - D_{\text{fade}}$$

$$D_{\text{Rotation}} = 45 \times D_{\text{fade}}$$

$$V_{\text{RotatedControl}} = Q_{\text{Calculate3DYaw}(\text{CurrentRotation})} \cdot \text{RotateVector}(V_{\text{PositionFeedbackControlOutput}})$$

$$D_{\text{TBX}} = D_{\text{TBX}} \times D_{\text{InverseFade}} + D_{\text{RotatedControlX}} \times D_{\text{Fade}} + D_{\text{RotatedControlZ}} \times D_{\text{Fade}}$$

$$D_{\text{TCX}} = D_{\text{TCX}} \times D_{\text{InverseFade}} + D_{\text{RotatedControlX}} \times D_{\text{Fade}} + D_{\text{RotatedControlZ}} \times D_{\text{Fade}}$$

$$D_{\text{TDX}} = D_{\text{TDX}} \times D_{\text{InverseFade}} + D_{\text{RotatedControlX}} \times D_{\text{Fade}} + D_{\text{RotatedControlZ}} \times D_{\text{Fade}}$$

$$D_{\text{TEX}} = D_{\text{TEX}} \times D_{\text{InverseFade}} + D_{\text{RotatedControlX}} \times D_{\text{Fade}} + D_{\text{RotatedControlZ}} \times D_{\text{Fade}}$$

$$D_{\text{TBZ}} = D_{\text{TBZ}} \times D_{\text{InverseFade}} + D_{\text{Rotation}}$$

$$D_{\text{TCZ}} = D_{\text{TCZ}} \times D_{\text{InverseFade}} - D_{\text{Rotation}}$$

$$D_{\text{TDZ}} = D_{\text{TDZ}} \times D_{\text{InverseFade}} + D_{\text{Rotation}}$$

$$D_{\text{TEZ}} = D_{\text{TEZ}} \times D_{\text{InverseFade}} - D_{\text{Rotation}}$$

$$D_{\text{PostionControlX}} = D_{\text{PostionControlX}} \times D_{\text{InverseFade}}$$

$$D_{\text{PostionControlZ}} = D_{\text{PostionControlZ}} \times D_{\text{InverseFade}}$$

20 ADRC

$$D_{\text{Amplification}}$$

$$D_{\text{Damping}}$$

$$D_{\text{Plant}}$$

$$P_{\text{PID}}$$

$$D_{\text{PrecisionModifier}}$$

$$D_{\text{Precision}} = D_{\text{TimeDerivative}} \times D_{\text{PrecisionModifier}}$$

$$O_{\text{CurrentOutput}} = (P_{\text{PID}} \cdot \text{Calculate}(D_{\text{SetPoint}}, D_{\text{ProcessVariable}}, D_{\text{TimeDerivate}}), O_{\text{PreviousOutput}})$$

$$S_{\text{State}} = ESO_{\text{ExtendedStateObserver}} \cdot \text{Observe}(D_{\text{TimeDerivative}}, O_{\text{CurrentOutput}}, D_{\text{Plant}}, D_{\text{ProcessVariable}})$$

$$D_{\text{PreviousOutput}} = D_{\text{CurrentOutput}}$$

$$D_{\text{CurrentOutput}} = NLC_{\text{NonlinearCombiner}} \cdot \text{Combine}(O_{\text{CurrentOutput}}, D_{\text{Plant}}, S_{\text{State}}, D_{\text{Precision}})$$

21 Setpoint Jump Prevention

$$D_{\text{Amp2Prec}} = D_{\text{Amplification}}^2 \times D_{\text{Precision}}$$

$$D_{\text{PrecTD}} = D_{\text{Precision}} \times D_{\text{TargetDerivative}}$$

$$D_{\text{TargetPrecTD}} = D_{\text{Target}} + D_{\text{PrecTD}}$$

$$D_{\text{A1}} = \sqrt{D_{\text{Amp2Prec}} \times (D_{\text{Amp2Prec}} + (8 \times |D_{\text{TargetPrecTD}}|))}$$

$$D_{\text{A2}} = \frac{D_{\text{PrecTD}} + \text{Sign}(D_{\text{TargetPrecTD}}) \times (D_{\text{A1}} - D_{\text{Amp2Prec}})}{2}$$

$$D_{\text{SignTargetPrecTD}} = \frac{\text{Sign}(D_{\text{TargetPrecTD}} + D_{\text{Amp2Prec}}) - \text{Sign}(D_{\text{TargetPrecTD}} - D_{\text{Amp2Prec}})}{2}$$

$$D_{\text{A}} = (D_{\text{PrecTD}} + D_{\text{TargetPrecTD}} - D_{\text{A2}}) \times D_{\text{SignTargetPrecTD}} + D_{\text{A2}}$$

$$D_{\text{SignA}} = \frac{\text{Sign}(D_{\text{A}} + D_{\text{Amp2Prec}}) - \text{Sign}(D_{\text{A}} - D_{\text{Amp2Prec}})}{2}$$

$$D_{\text{SetpointJumpPrevention}} = -D_{\text{Amplification}} \times \left(\frac{D_{\text{A}}}{D_{\text{Amp2Prec}}} \right) - \text{Sign}(D_{\text{A}}) \times D_{\text{SignA}} - D_{\text{Amplification}} \times \text{Sign}(D_{\text{A}})$$

22 Nonlinear Combiner

$$D_{\text{EstimationE1}} = O_{\text{CurrentOutput}} - S_{\text{StateZ1}}$$

$$D_{\text{EstimationE2}} = O_{\text{PreviousOutput}} - S_{\text{StateZ2}}$$

$$D_{\text{NominalControlSignal}} = \text{SetPointJumpPrevention}(D_{\text{Target}}, D_{\text{TargetDerivate}}, D_{\text{Amplification}}, D_{\text{Precision}})$$

$$D_{\text{Combine}} = \frac{D_{\text{NominalControlSignal}} + S_{\text{StateZ3}}}{D_{\text{Plant}}}$$

23 Extended State Observer

$$S_{\text{Gain1}} = 1$$

$$S_{\text{Gain2}} = \frac{1}{2 \times dT^{0.5}}$$

$$S_{\text{Gain3}} = \frac{2}{25 \times dT^{1.2}}$$

$$D_{\text{E}} = D_{\text{StateZ1}} - D_{\text{ProcessVariable}}$$

$$D_{\text{FE}} = \text{NonlinearFunction}(D_{\text{E}}, 0.5, dT)$$

$$D_{\text{FE1}} = \text{NonlinearFunction}(D_{\text{E}}, 0.25, dT)$$

$$S_{\text{StateZ1}} = S_{\text{StateZ1}} + (dT \times S_{\text{StateZ2}}) - (S_{\text{GainZ1}} \times D_{\text{E}})$$

$$S_{\text{StateZ2}} = S_{\text{StateZ2}} + (dT \times (S_{\text{StateZ3}} + (D_{\text{Plant}} \times D_{\text{Output}}))) - (S_{\text{GainZ2}} \times D_{\text{FE}})$$

$$S_{\text{StateZ3}} = S_{\text{StateZ3}} - S_{\text{GainZ3}} \times D_{\text{FE1}}$$

24 Nonlinear Function

$$\text{Output} = \begin{cases} \frac{\eta}{\delta^{(1-\alpha)}}, & \text{if } |\eta| \leq \delta. \\ |\eta|^\alpha \times \text{Sign}(\eta), & \text{otherwise.} \end{cases}$$

25 Axis-Angle to Quaternion

$$D_{\text{Rotation}} = \frac{AA_{\text{AxisAngleRotation}} \times \pi}{180}$$

$$D_{\text{Scale}} = \sin\left(\frac{D_{\text{Rotation}}}{2}\right)$$

$$Q_{\text{W}} = \cos\left(\frac{D_{\text{Rotation}}}{2}\right)$$

$$Q_{\text{X}} = AA_{\text{AxisAngleX}} \times D_{\text{Scale}}$$

$$Q_{\text{Y}} = AA_{\text{AxisAngleY}} \times D_{\text{Scale}}$$

$$Q_{\text{Z}} = AA_{\text{AxisAngleZ}} \times D_{\text{Scale}}$$

26 Quaternion to Axis-Angle

$$AA_{\text{Rotation}} = 2 \times \text{acos}(Q_W)$$

$$D_{\text{QuaternionCheck}} = \sqrt{1 - Q_W^2}$$

$$AA_{\text{AxisX}} = \begin{cases} \frac{Q_X}{D_{\text{QuaternionCheck}}}, & \text{if } D_{\text{QuaternionCheck}} \geq 0.001. \\ 0, & \text{otherwise.} \end{cases}$$

$$AA_{\text{AxisY}} = \begin{cases} \frac{Q_Y}{D_{\text{QuaternionCheck}}}, & \text{if } D_{\text{QuaternionCheck}} \geq 0.001. \\ 0, & \text{otherwise.} \end{cases}$$

$$AA_{\text{AxisZ}} = \begin{cases} \frac{Q_Z}{D_{\text{QuaternionCheck}}}, & \text{if } D_{\text{QuaternionCheck}} \geq 0.001. \\ 0, & \text{otherwise.} \end{cases}$$

27 Direction-Angle to Quaternion

$$V_{\text{Right}} = \text{Vector}(1, 0, 0)$$

$$V_{\text{Up}} = \text{Vector}(0, 1, 0)$$

$$V_{\text{Forward}} = \text{Vector}(0, 0, 1)$$

$$V_{\text{RotatedUp}} = DA_{\text{DirectionAngleDirection}}$$

$$Q_{\text{Rotation}} = \text{QuaternionFromDirectionVectors}(V_{\text{Up}}, V_{\text{RotatedUp}})$$

$$V_{\text{RotatedRight}} = RM_{\text{RotationMatrix}}.\text{Rotate}(\text{Vector}(0, -DA_{\text{DirectionAngleRotation}}, 0), V_{\text{Right}})$$

$$V_{\text{RotatedForward}} = RM_{\text{RotationMatrix}}.\text{Rotate}(\text{Vector}(0, -DA_{\text{DirectionAngleRotation}}, 0), V_{\text{Forward}})$$

$$V_{\text{RotatedRight}} = Q_{\text{Rotation}}.\text{RotateVector}(V_{\text{RotatedRight}})$$

$$V_{\text{RotatedForward}} = Q_{\text{Rotation}}.\text{RotateVector}(V_{\text{RotatedForward}})$$

$$Q_{\text{Quaternion}} = RMT\text{ToQuaternion}(\text{RotationMatrix}(V_{\text{RotatedRight}}, V_{\text{RotatedUp}}, V_{\text{RotatedForward}}))$$

28 Quaternion to Direction-Angle

$$V_{Up} = Vector(0, 1, 0)$$

$$V_{Right} = Vector(1, 0, 0)$$

$$V_{RotatedUp} = Q_{Quaternion}.RotateVector(V_{Up})$$

$$V_{RotatedRight} = Q_{Quaternion}.RotateVector(V_{Right})$$

$$Q_{Rotation} = QuaternionFromDirectionVectors(V_{Up}, V_{RotateUp})$$

$$V_{RightCompensated} = Q_{Rotation}.UnrotateVector(V_{RotatedRight})$$

$$D_{RightAngle} = \frac{atan2(D_{RightZ}, D_{RightX}) \times \pi}{180}$$

$$D_{RightRotatedAngle} = \frac{atan2(D_{RightCompensatedZ}, D_{RightCompensatedX}) \times \pi}{180}$$

$$DA_{Rotation} = D_{RightAngle} - D_{RightRotateAngle}$$

$$DA_{Direction} = V_{RotatedUp}$$

29 Rotation Matrix to Quaternion

$$D_{\text{MatrixTrace}} = V_{\text{XAxisX}} + V_{\text{YAxisY}} + V_{\text{ZAxisZ}}$$

$$D_{\text{Square}} = \begin{cases} \sqrt{1 + \text{MatrixTrace}} \times 2, & \text{if } D_{\text{MatrixTrace}} > 0. \\ \sqrt{1 + D_{\text{XAxisX}} - D_{\text{YAxisY}} - D_{\text{ZAxisZ}}} \times 2, & \text{if } D_{\text{XAxisX}} > D_{\text{YAxisY}} \text{ and } D_{\text{XAxisX}} > D_{\text{ZAxisZ}}. \\ \sqrt{1 + D_{\text{YAxisY}} - D_{\text{XAxisX}} - D_{\text{ZAxisZ}}} \times 2, & \text{if } D_{\text{YAxisY}} > D_{\text{ZAxisZ}}. \\ \sqrt{1 + D_{\text{ZAxisZ}} - D_{\text{XAxisX}} - D_{\text{YAxisY}}} \times 2, & \text{otherwise.} \end{cases}$$

$$Q_W = \begin{cases} \frac{D_{\text{Square}}}{4}, & \text{if } D_{\text{MatrixTrace}} > 0. \\ \frac{D_{\text{ZAxisY}} - D_{\text{YAxisZ}}}{D_{\text{Square}}}, & \text{if } D_{\text{XAxisX}} > D_{\text{YAxisY}} \text{ and } D_{\text{XAxisX}} > D_{\text{ZAxisZ}}. \\ \frac{D_{\text{XAxisZ}} - D_{\text{ZAxisX}}}{D_{\text{Square}}}, & \text{if } D_{\text{YAxisY}} > D_{\text{ZAxisZ}}. \\ \frac{D_{\text{YAxisX}} - D_{\text{XAxisY}}}{D_{\text{Square}}}, & \text{otherwise.} \end{cases}$$

$$Q_X = \begin{cases} \frac{D_{\text{ZAxisY}} - D_{\text{YAxisZ}}}{D_{\text{Square}}}, & \text{if } D_{\text{MatrixTrace}} > 0. \\ \frac{D_{\text{Square}}}{4}, & \text{if } D_{\text{XAxisX}} > D_{\text{YAxisY}} \text{ and } D_{\text{XAxisX}} > D_{\text{ZAxisZ}}. \\ \frac{D_{\text{XAxisY}} - D_{\text{YAxisX}}}{D_{\text{Square}}}, & \text{if } D_{\text{YAxisY}} > D_{\text{ZAxisZ}}. \\ \frac{D_{\text{XAxisZ}} - D_{\text{ZAxisX}}}{D_{\text{Square}}}, & \text{otherwise.} \end{cases}$$

$$Q_Y = \begin{cases} \frac{D_{\text{XAxisZ}} - D_{\text{ZAxisX}}}{D_{\text{Square}}}, & \text{if } D_{\text{MatrixTrace}} > 0. \\ \frac{D_{\text{XAxisY}} - D_{\text{YAxisX}}}{D_{\text{Square}}}, & \text{if } D_{\text{XAxisX}} > D_{\text{YAxisY}} \text{ and } D_{\text{XAxisX}} > D_{\text{ZAxisZ}}. \\ \frac{D_{\text{Square}}}{4}, & \text{if } D_{\text{YAxisY}} > D_{\text{ZAxisZ}}. \\ \frac{D_{\text{YAxisZ}} - D_{\text{ZAxisY}}}{D_{\text{Square}}}, & \text{otherwise.} \end{cases}$$

$$Q_Z = \begin{cases} \frac{D_{\text{YAxisZ}} - D_{\text{ZAxisY}}}{D_{\text{Square}}}, & \text{if } D_{\text{MatrixTrace}} > 0. \\ \frac{D_{\text{XAxisZ}} - D_{\text{ZAxisX}}}{D_{\text{Square}}}, & \text{if } D_{\text{XAxisX}} > D_{\text{YAxisY}} \text{ and } D_{\text{XAxisX}} > D_{\text{ZAxisZ}}. \\ \frac{D_{\text{YAxisZ}} - D_{\text{ZAxisY}}}{D_{\text{Square}}}, & \text{if } D_{\text{YAxisY}} > D_{\text{ZAxisZ}}. \\ \frac{D_{\text{Square}}}{4}, & \text{otherwise.} \end{cases}$$

$$Q_{\text{WXYZ}} = \text{Conjugate}(Q_{\text{WXYZ}})$$

30 Quaternion to Rotation Matrix

$$V_{\text{Right}} = \text{Vector}(1, 0, 0)$$

$$V_{\text{Up}} = \text{Vector}(0, 1, 0)$$

$$V_{\text{Forward}} = \text{Vector}(0, 0, 1)$$

$$RM_{\text{XAxis}} = Q_{\text{Rotation}} \cdot \text{RotateVector}(V_{\text{Right}})$$

$$RM_{\text{YAxis}} = Q_{\text{Rotation}} \cdot \text{RotateVector}(V_{\text{Up}})$$

$$RM_{\text{ZAxis}} = Q_{\text{Rotation}} \cdot \text{RotateVector}(V_{\text{Forward}})$$

31 Direction Vectors to Quaternion

$$V_{XAxis} = Vector(1, 0, 0)$$

$$V_{YAxis} = Vector(0, 1, 0)$$

$$D_{Dot} = V_{Initial} \cdot V_{Final}$$

$$C_{Cross} = \begin{cases} V_{XAxis} \times V_{Initial}, & \text{if } D_{Dot} < -0.999. \\ Vector(0, 0, 0), & \text{if } D_{Dot} > 0.999. \\ V_{Initial} \times V_{Final}, & \text{otherwise.} \end{cases}$$

$$C_{Cross} = \begin{cases} V_{YAxis} \times V_{Initial}, & \text{if } D_{CrossLength} < 0.001. \end{cases}$$

$$AA_{AxisAngle} = AxisAngle(\pi, V_{Cross})$$

$$Q_W = \begin{cases} D_{AxisAngleRotation}, & \text{if } D_{Dot} < -0.999. \\ 1, & \text{if } D_{Dot} > 0.999. \\ 1 + D_{Dot}, & \text{otherwise.} \end{cases}$$

$$Q_X = \begin{cases} D_{AxisAngleAxisX}, & \text{if } D_{Dot} < -0.999. \\ 0, & \text{if } D_{Dot} > 0.999. \\ D_{CrossX}, & \text{otherwise.} \end{cases}$$

$$Q_Y = \begin{cases} D_{AxisAngleAxisY}, & \text{if } D_{Dot} < -0.999. \\ 0, & \text{if } D_{Dot} > 0.999. \\ D_{CrossY}, & \text{otherwise.} \end{cases}$$

$$Q_Z = \begin{cases} D_{AxisAngleAxisZ}, & \text{if } D_{Dot} < -0.999. \\ 0, & \text{if } D_{Dot} > 0.999. \\ D_{CrossZ}, & \text{otherwise.} \end{cases}$$

32 Euler Angles to Quaternion

$$D_{\text{Rotating}} = D_{\text{EulerX}}$$

$$D_{\text{EulerX}} = \begin{cases} D_{\text{EulerZ}}, & \text{if } F_{\text{Frame}} = F_{\text{Rotating}} \\ D_{\text{EulerX}}, & \text{otherwise.} \end{cases}$$

$$D_{\text{EulerZ}} = \begin{cases} D_{\text{Rotating}}, & \text{if } F_{\text{Frame}} = F_{\text{Rotating}} \\ D_{\text{EulerZ}}, & \text{otherwise.} \end{cases}$$

$$D_{\text{EulerY}} = \begin{cases} -D_{\text{EulerY}}, & \text{if } P_{\text{AxisPermutation}} = P_{\text{Odd}} \\ D_{\text{EulerY}}, & \text{otherwise.} \end{cases}$$

$$D_{\text{SineX}} = \sin\left(\frac{D_{\text{EulerX}}}{2}\right)$$

$$D_{\text{SineY}} = \sin\left(\frac{D_{\text{EulerY}}}{2}\right)$$

$$D_{\text{SineZ}} = \sin\left(\frac{D_{\text{EulerZ}}}{2}\right)$$

$$D_{\text{CosineX}} = \cos\left(\frac{D_{\text{EulerX}}}{2}\right)$$

$$D_{\text{CosineY}} = \cos\left(\frac{D_{\text{EulerY}}}{2}\right)$$

$$D_{\text{CosineZ}} = \cos\left(\frac{D_{\text{EulerZ}}}{2}\right)$$

$$D_{\text{CC}} = D_{\text{CosineX}} \times D_{\text{CosineZ}}$$

$$D_{\text{CS}} = D_{\text{CosineX}} \times D_{\text{SineZ}}$$

$$D_{\text{SC}} = D_{\text{SineX}} \times D_{\text{CosineZ}}$$

$$D_{\text{SS}} = D_{\text{SineX}} \times D_{\text{SineZ}}$$

$$Q_{\text{W}} = \begin{cases} D_{\text{CosineY}} \times (D_{\text{CC}} - D_{\text{SS}}), & \text{if } A_{\text{AxisRepetition}} = A_{\text{Yes}} \\ D_{\text{CosineY}} \times D_{\text{CC}} + D_{\text{SineY}} \times D_{\text{SS}}, & \text{otherwise.} \end{cases}$$

$$Q_{\text{X}} = \begin{cases} D_{\text{CosineY}} \times (D_{\text{CS}} + D_{\text{SC}}), & \text{if } A_{\text{AxisRepetition}} = A_{\text{Yes}} \\ D_{\text{CosineY}} \times D_{\text{SC}} - D_{\text{SineY}} \times D_{\text{CS}}, & \text{otherwise.} \end{cases}$$

$$Q_{\text{Y}} = \begin{cases} D_{\text{SineY}} \times (D_{\text{CC}} + D_{\text{SS}}), & \text{if } A_{\text{AxisRepetition}} = A_{\text{Yes}} \\ D_{\text{CosineY}} \times D_{\text{SS}} + D_{\text{SineY}} \times D_{\text{CC}}, & \text{otherwise.} \end{cases}$$

$$Q_{\text{Z}} = \begin{cases} D_{\text{SineY}} \times (D_{\text{CS}} - D_{\text{SC}}), & \text{if } A_{\text{AxisRepetition}} = A_{\text{Yes}} \\ D_{\text{CosineY}} \times D_{\text{CS}} - D_{\text{SineY}} \times D_{\text{SC}}, & \text{otherwise.} \end{cases}$$

$$Q_{\text{WXYZ}} = Q_{\text{WXYZ}}.Permutate(V_{\text{Permutation}})$$

$$Q_{\text{Y}} = \begin{cases} -Q_{\text{Y}}, & \text{if } P_{\text{AxisPermutation}} = P_{\text{Odd}} \\ Q_{\text{Y}}, & \text{otherwise.} \end{cases}$$

33 Quaternion to Euler Angles

$$D_{\text{Norm}} = D_{\text{QuaternionNormal}}$$

$$D_{\text{Scale}} = \begin{cases} \frac{2}{D_{\text{Norm}}}, & \text{if } D_{\text{Norm}} > 0 \\ 0, & \text{otherwise.} \end{cases}$$

$$V_{\text{SB}} = \text{Vector}(Q_X \times D_{\text{Scale}}, Q_Y \times D_{\text{Scale}}, Q_Z \times D_{\text{Scale}})$$

$$V_{\text{W}} = \text{Vector}(Q_{\text{W}} \times SB_X, Q_{\text{W}} \times SB_Y, Q_{\text{W}} \times SB_Z)$$

$$V_X = \text{Vector}(Q_X \times SB_X, Q_X \times SB_Y, Q_X \times SB_Z)$$

$$V_Y = \text{Vector}(0, Q_Y \times SB_Y, Q_Y \times SB_Z)$$

$$V_Z = \text{Vector}(0, 0, Q_Z \times SB_Z)$$

$$V_{\text{XAxis}} = \text{Vector}(1 - D_{YY} + D_{ZZ}, D_{XY} - D_{WZ}, D_{XZ} + D_{WY})$$

$$V_{\text{YAxis}} = \text{Vector}(D_{XY} + D_{WZ}, 1 - D_{XX} - D_{ZZ}, D_{YZ} - D_{WX})$$

$$V_{\text{ZAxis}} = \text{Vector}(D_{XZ} - D_{WY}, D_{YZ} - D_{WX}, 1 - D_{XX} + D_{YY})$$

$$RM_{\text{RotationMatrix}} = \text{RotationMatrix}(V_{\text{XAxis}}, V_{\text{YAxis}}, V_{\text{ZAxis}})$$

$$D_{\text{SineY}} = \sqrt{(D_{XY}^2) + (D_{XZ}^2)}$$

$$D_{\text{CosineY}} = \sqrt{(D_{XX}^2) + (D_{YY}^2)}$$

$$D_{\text{EAX}} = \begin{cases} \text{atan2}(D_{XY}, D_{XZ}), & \text{if Initial Axis is repeated and } D_{\text{SineY}} > 32 \times D_{\text{Epsilon}} \\ \text{atan2}(-D_{YZ}, D_{YY}), & \text{if Initial Axis is repeated and } D_{\text{SineY}} \leq 32 \times D_{\text{Epsilon}} \\ \text{atan2}(D_{ZY}, D_{ZZ}), & \text{if Initial Axis is not repeated and } D_{\text{SineY}} > 32 \times D_{\text{Epsilon}} \\ \text{atan2}(-D_{YZ}, D_{YY}), & \text{if Initial Axis is not repeated and } D_{\text{SineY}} \leq 32 \times D_{\text{Epsilon}} \end{cases}$$

$$D_{\text{EAY}} = \begin{cases} \text{atan2}(D_{\text{SineY}}, D_{XX}), & \text{if Initial Axis is repeated} \\ \text{atan2}(-D_{ZX}, D_{\text{CosineY}}), & \text{if Initial Axis is not repeated} \end{cases}$$

$$\begin{aligned}
D_{\text{EAX}} &= \begin{cases} \text{atan2}(D_{\text{YX}}, -D_{\text{ZX}}), & \text{if Initial Axis is repeated and } D_{\text{SineY}} > 32 \times D_{\text{Epsilon}} \\ 0, & \text{if Initial Axis is repeated and } D_{\text{SineY}} \leq 32 \times D_{\text{Epsilon}} \\ \text{atan2}(D_{\text{YX}}, D_{\text{XX}}), & \text{if Initial Axis is not repeated and } D_{\text{SineY}} > 32 \times D_{\text{Epsilon}} \\ 0, & \text{if Initial Axis is not repeated and } D_{\text{SineY}} \leq 32 \times D_{\text{Epsilon}} \end{cases} \\
D_{\text{EAX}} &= \begin{cases} -D_{\text{EAX}}, & \text{if Axis Permutation is Odd} \end{cases} \\
D_{\text{EAY}} &= \begin{cases} -D_{\text{EAY}}, & \text{if Axis Permutation is Odd} \end{cases} \\
D_{\text{EAZ}} &= \begin{cases} -D_{\text{EAX}}, & \text{if Axis Permutation is Odd} \end{cases} \\
D_{\text{TempX}} &= D_{\text{EAX}} \\
D_{\text{EAX}} &= \begin{cases} D_{\text{EAX}}, & \text{if Frame Taken is Rotating} \end{cases} \\
D_{\text{EAX}} &= \begin{cases} D_{\text{TempX}}, & \text{if Frame Taken is Rotating} \end{cases}
\end{aligned}$$

34 Kalman Filter

ArrayDataset.Add(D_{Datapoint})

if $I_{\text{DatasetLength}} > I_{\text{Memory}}$, *ArrayDataset.RemoveLast()*

$D_{\text{Sum}} = \sum \text{ArrayDataset}$

$D_{\text{Average}} = \frac{D_{\text{Sum}}}{I_{\text{Memory}}}$

$D_{\text{FilteredValue}} = D_{\text{KalmanGain}} \times D_{\text{Datapoint}} + (1 - D_{\text{KalmanGain}}) \times D_{\text{Average}}$

35 Quaternion Kalman Filter

ArrayDataset.Add(Q_{Datapoint})

if $I_{\text{DatasetLength}} > I_{\text{Memory}}$, *ArrayDataset.RemoveLast()*

$Q_{\text{Sum}} = \sum \text{ArrayDataset}$

$Q_{\text{Average}} = \frac{Q_{\text{Sum}}}{I_{\text{Memory}}}$

$D_{\text{FilteredValue}} = \text{SphericalInterpolation}(Q_{\text{Datapoint}}, Q_{\text{Average}}, 1 - D_{\text{KalmanGain}})$

36 Finite Impulse Response Filter

$$\lambda = \frac{\pi \times D_{\text{CutFrequency}} \times 2}{D_{\text{SamplingRate}}}$$

$$\phi = \frac{\pi \times D_{\text{SecondaryCutFrequency}} \times 2}{D_{\text{SamplingRate}}}$$