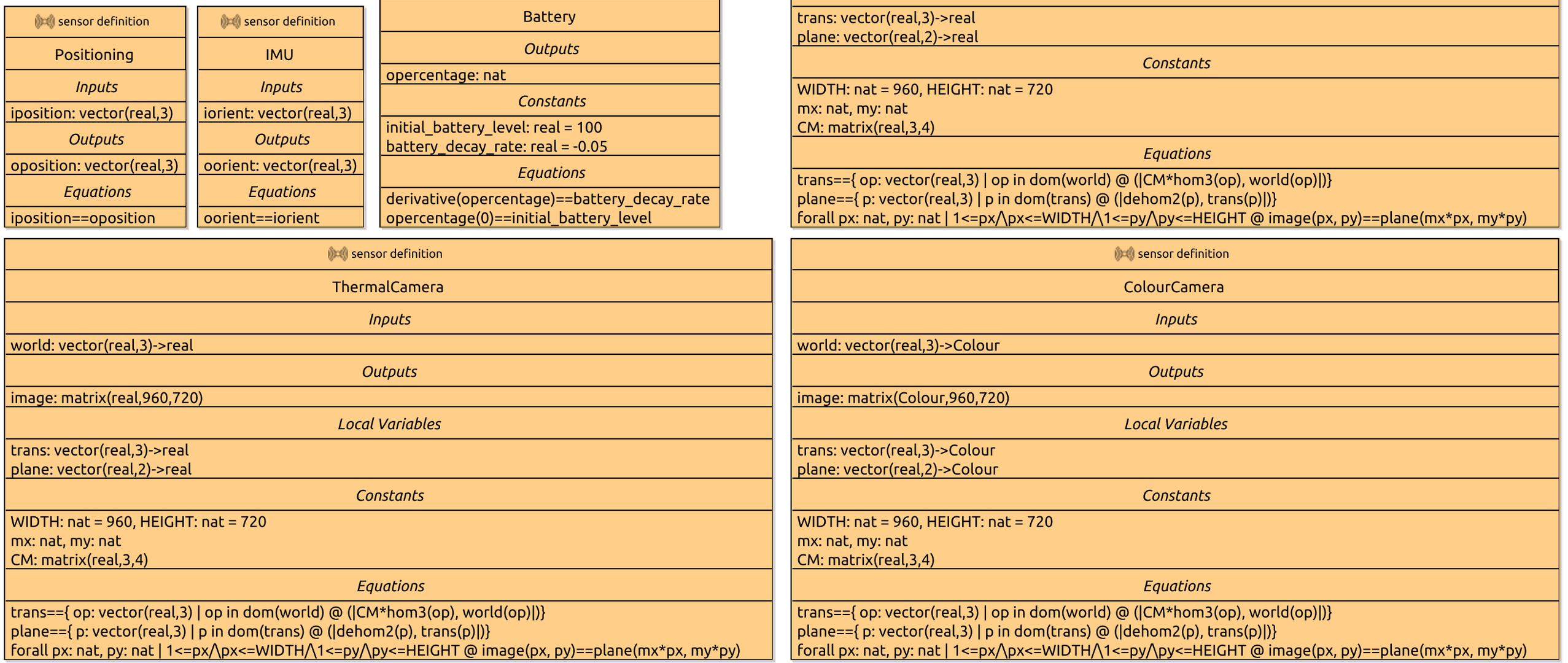
sensor definition
Positioning
Inputs
iposition: vector(real,3)
Outputs
oposition: vector(real,3)
Equations
iposition==oposition
sensor definition
IMU
Inputs
iorient: vector(real,3)
Outputs
oorient: vector(real,3)
Equations
oorient==iorient
sensor definition
Battery
Outputs
opercentage: nat
Constants
initial_battery_level: real = 100
battery_decay_rate: real = -0.05
Equations
derivative(opercentage)==battery_decay_rate
opercentage(0)==initial_battery_level
sensor definition
DepthCamera
Inputs
world: vector(real,3)->real
Outputs
image: matrix(real,960,720)
Local Variables
trans: vector(real,3)->real
plane: vector(real,2)->real
Constants
WIDTH: nat = 960, HEIGHT: nat = 720
mx: nat, my: nat
CM: matrix(real,3,4)
Equations
trans=={ op: vector(real,3) | op in dom(world) @ (|CM*hom3(op), world(op)|)}
plane=={ p: vector(real,3) | p in dom(trans) @ (|dehom2(p), trans(p)|)}
forall px: nat, py: nat | 1<=px/\px<=WIDTH/\1<=py/\py<=HEIGHT @ image(px, py)==plane(mx*px, my*py)
sensor definition
ThermalCamera
Inputs
world: vector(real,3)->real
Outputs
image: matrix(real,960,720)
Local Variables
trans: vector(real,3)->real
plane: vector(real,2)->real
Constants
WIDTH: nat = 960, HEIGHT: nat = 720
mx: nat, my: nat
CM: matrix(real,3,4)
Equations
trans=={ op: vector(real,3) | op in dom(world) @ (|CM*hom3(op), world(op)|)}
plane=={ p: vector(real,3) | p in dom(trans) @ (|dehom2(p), trans(p)|)}
forall px: nat, py: nat | 1<=px/\px<=WIDTH/\1<=py/\py<=HEIGHT @ image(px, py)==plane(mx*px, my*py)
sensor definition
ColourCamera
Inputs
world: vector(real,3)->Colour
Outputs
image: matrix(Colour,960,720)
Local Variables
trans: vector(real,3)->Colour
plane: vector(real,2)->Colour
Constants
WIDTH: nat = 960, HEIGHT: nat = 720
mx: nat, my: nat
CM: matrix(real,3,4)
Equations
trans=={ op: vector(real,3) | op in dom(world) @ (|CM*hom3(op), world(op)|)}
plane=={ p: vector(real,3) | p in dom(trans) @ (|dehom2(p), trans(p)|)}
forall px: nat, py: nat | 1<=px/\px<=WIDTH/\1<=py/\py<=HEIGHT @ image(px, py)==plane(mx*px, my*py)