In[10]:=

```
(*Define synthetic stereo setup and computation of Fundamentalmatrix*)
(*_______________________________________________________________*)

(*Moduls for camera orientation___________________________*)
RotationsE1[alpha_] := Module[{Re1},
                       Modul
   Re1 = {{1, 0, 0}, {0, Cos[alpha Degree], -Sin[alpha Degree]},
                         Kosinus  Grad        Sinus  Grad
     {0, Sin[alpha Degree], Cos[alpha Degree]}};
        Sinus  Grad         Kosinus  Grad
   Return[Re1]
   gib zurück
  ];

RotationsE2[alpha_] := Module[{Re2},
                       Modul
   Re2 = {{Cos[alpha Degree], 0, Sin[alpha Degree]},
           Kosinus  Grad        Sinus  Grad
     {0, 1, 0}, {-Sin[alpha Degree], 0, Cos[alpha Degree]}};
                 Sinus  Grad           Kosinus  Grad
   Return[Simplify[Re2]]
   gib zur··· vereinfache
  ];

RotationsE3[alpha_] := Module[{Re3},
                       Modul
   Re3 = {{Cos[alpha Degree], -Sin[alpha Degree], 0},
           Kosinus  Grad        Sinus  Grad
     {Sin[alpha Degree], Cos[alpha Degree], 0}, {0, 0, 1}};
      Sinus  Grad        Kosinus  Grad
   Return[Re3]
   gib zurück
  ];

(*Start of Computation__________________________________*)
StartComputation[] := Module[{},
                      Modul

   ComputeTranslationAndRotation[Rot1, Rot2, Oc2, zeta2];
  ];

(*Place Cameras in 3D Scene______________________________*)
                   leite ab
ComputeTranslationAndRotation[Rotation_, Rotations2_, Oc2_, zeta2_] :=
  Module[{V, V2, M, PM1, PM2, R, R2, M2},
  Modul
   PM2 = {{zeta2, 0, 0, 0}, {0, zeta2, 0, 0}, {0, 0, zeta2, 0}, {0, 0, 1, 0}};
   PM1 = {{zeta1, 0, 0, 0}, {0, zeta1, 0, 0}, {0, 0, zeta1, 0}, {0, 0, 1, 0}};

   Print["PM1 = ", PM1];
   gib aus
   Print["PM2 = ", PM2];
   gib aus
   PM2 = A.PM2;
   Print["PM2 = ", PM2];
   ⌊gib aus

   R = Transpose[Rotation];
      ⌊transponiere
   R2 = Transpose[Rotations2];
       ⌊transponiere

   Print["OC2 = ", Oc2];
   ⌊gib aus
   V = {{1, 0, 0, -Oc[[1]]}, {0, 1, 0, -Oc[[2]]}, {0, 0, 1, -Oc[[3]]}};
   M = R.V;
   M = {{M[[1, 1]], M[[1, 2]], M[[1, 3]], M[[1, 4]]}, {M[[2, 1]], M[[2, 2]], M[[2, 3]],
      M[[2, 4]]}, {M[[3, 1]], M[[3, 2]], M[[3, 3]], M[[3, 4]]}, {0, 0, 0, 1}};


   V2 = {{1, 0, 0, -Oc2[[1]]}, {0, 1, 0, -Oc2[[2]]}, {0, 0, 1, -Oc2[[3]]}};
   M2 = R2.V2;
   M2 = {{M2[[1, 1]], M2[[1, 2]], M2[[1, 3]], M2[[1, 4]]},
     {M2[[2, 1]], M2[[2, 2]], M2[[2, 3]], M2[[2, 4]]},
     {M2[[3, 1]], M2[[3, 2]], M2[[3, 3]], M2[[3, 4]]}, {0, 0, 0, 1}};


   Print["M of C2=", MatrixForm[Simplify[M]]];
   ⌊gib aus          ⌊Matritzenform ⌊vereinfache
   Print["M2 of C1 =", MatrixForm[N[M2]]];
   ⌊gib aus            ⌊Matritzenform ⌊numerischer Wert
   ComputeProjectionMtx[a, b, c, d, aPrime,
    bPrime, cPrime, dPrime, d2Prime, Oc, Oc2, M, PM1, PM2, M2];
  ];

(*Compute Projectionmatrices________________________________*)
ComputeProjectionMtx[a_, b_, c_, d_, aPrime_, bPrime_,
   cPrime_, dPrime_, d2Prime_, Oc_, Oc2_, M_, PM1_, PM2_, M2_] :=
  Module [{t, ProjectionMtxCamera1, ProjectionMtxCamera2},
  ⌊Modul
   ProjectionMtxCamera1  = PM1.M;
   ProjectionMtxCamera2 = PM2.M2;

   Print["ProjectionMtxCamera1", MatrixForm[ProjectionMtxCamera1]];
   ⌊gib aus                     ⌊Matritzenform
   Print["ProjectionMtxCamera2", MatrixForm[N[ProjectionMtxCamera2]]];
   ⌊gib aus                     ⌊Matritzenform ⌊numerischer Wert
   ComputeProjectedPointsCamera1And2[ProjectionMtxCamera1, ProjectionMtxCamera2];

  ];

(*Project Points to Imageplanes in cameracoordinates_________*)
ComputeProjectedPointsCamera1And2[ProjectionMtxCamera1_, ProjectionMtxCamera2_] :=
  Module[{CameraProjectedPointsK1, CameraProjectedPointsK2,
  ⌊Modul
    GraphicPointsC1, GraphicPointsC2, G1, G2},
CameraProjectedPointsK1 = Map[ProjectionMtxCamera1.# &,
                          wende an
  {a, b, c, d, aPrime, bPrime, cPrime, dPrime, d2Prime}];
CameraProjectedPointsK2 = Map[ProjectionMtxCamera2.# &,
                          wende an
  {a, b, c, d, aPrime, bPrime, cPrime, dPrime, d2Prime}];

Print["CameraProjectedPointsK1 = ", MatrixForm[CameraProjectedPointsK1]];
gib aus                             Matritzenform
Print["CameraProjectedPointsK2 = ", MatrixForm[N[CameraProjectedPointsK2]]];
gib aus                             Matritzenform numerischer Wert

For[i = 1, i ≤ 9, i++,
For-Schleife
 CameraProjectedPointsK1[[i]] =
  CameraProjectedPointsK1[[i]] / CameraProjectedPointsK1[[i, 4]];
 CameraProjectedPointsK2[[i]] = CameraProjectedPointsK2[[i]] /
   CameraProjectedPointsK2[[i, 4]];

];

Print["homogene CameraProjectedPointsK1 = ",
gib aus
 MatrixForm[CameraProjectedPointsK1]];
 Matritzenform
Print["homogene CameraProjectedPointsK2 = ",
gib aus
 MatrixForm[N[CameraProjectedPointsK2]]];
 Matritzenform numerischer Wert

Print[
gib aus
 "Begin construct Epipol_______________________________________________"];
  beginne Kontext
if[alpha == 45, ConstructEpipole[Oc, Oc2, RotationsE2[alpha],
  CameraProjectedPointsK2[[1]], ProjectionMtxCamera2]];

Print["End construct Epipol_______________________________________________"];
gib aus beende Kontext

GraphicPointsC1 = Map[{#[[1]], #[[2]]} &, CameraProjectedPointsK1];
                  wende an
GraphicPointsC2 = Map[{#[[1]], #[[2]]} &, CameraProjectedPointsK2];
                  wende an
G1 = Show[ListPlot[GraphicPointsC1[[1 ;; 9]], PlotStyle → Darker[Green]],
     zeig··· listenbezogene Graphik                Darstellungsstil dunkler grün
  ListLinePlot[{GraphicPointsC1[[4, All]], GraphicPointsC1[[1, All]],
  listenbezogene Liniengraphik      alle                       alle
    GraphicPointsC1[[2, All]], GraphicPointsC1[[3, All]],
                        alle                       alle
    GraphicPointsC1[[4, All]], GraphicPointsC1[[8, All]],
                        alle                       alle
    GraphicPointsC1[[7, All]], GraphicPointsC1[[6, All]], GraphicPointsC1[[5
                        alle                       alle
```

```
                          alle                    alle
        , All]], GraphicPointsC1[[8, All]]}, PlotStyle → Darker[Green]],
         alle                     alle      Darstellungsstil dunkler grün
    ListLinePlot[{GraphicPointsC1[[1, All]], GraphicPointsC1[[5, All]]},
    listenbezogene Liniengraphik    alle                     alle
     PlotStyle → Darker[Green]],
     Darstellungsstil dunkler grün
    ListLinePlot[{GraphicPointsC1[[2, All]], GraphicPointsC1[[6, All]]},
    listenbezogene Liniengraphik    alle                     alle
     PlotStyle → Darker[Green]],
     Darstellungsstil dunkler grün
    ListLinePlot[{GraphicPointsC1[[3, All]], GraphicPointsC1[[7, All]]},
    listenbezogene Liniengraphik    alle                     alle
     PlotStyle → Darker[Green]]];
     Darstellungsstil dunkler grün
  G2 = Show[ListPlot[GraphicPointsC2[[1 ;; 9]], PlotStyle → Darker[Red]],
       zeig··· listenbezogene Graphik              Darstellungsstil dunkler rot
    ListLinePlot[{GraphicPointsC2[[4, All]], GraphicPointsC2[[1, All]],
    listenbezogene Liniengraphik    alle                     alle
      GraphicPointsC2[[2, All]], GraphicPointsC2[[3, All]],
                          alle                    alle
      GraphicPointsC2[[4, All]], GraphicPointsC2[[8, All]],
                          alle                    alle
      GraphicPointsC2[[7, All]], GraphicPointsC2[[6, All]], GraphicPointsC2[[5
                          alle                    alle
        , All]], GraphicPointsC2[[8, All]]}, PlotStyle → Darker[Red]],
         alle                     alle      Darstellungsstil dunkler rot
    ListLinePlot[{GraphicPointsC2[[1, All]], GraphicPointsC2[[5, All]]},
    listenbezogene Liniengraphik    alle                     alle
     PlotStyle → Darker[Red]],
     Darstellungsstil dunkler rot
    ListLinePlot[{GraphicPointsC2[[2, All]], GraphicPointsC2[[6, All]]},
    listenbezogene Liniengraphik    alle                     alle
     PlotStyle → Darker[Red]],
     Darstellungsstil dunkler rot
    ListLinePlot[{GraphicPointsC2[[3, All]], GraphicPointsC2[[7, All]]},
    listenbezogene Liniengraphik    alle                     alle
     PlotStyle → Darker[Red]]];
     Darstellungsstil dunkler rot
  Print[Show[G1, G2, PlotRange → All]];
  gib aus zeige an    Koordinatenb··· alle
  ComputeProjectedPointsOnImagePlane1And2[
   CameraProjectedPointsK1, CameraProjectedPointsK2, G1, G2];
 ];


(*Norm Points to Imageplanes______________________________*)
 Norm
ComputeProjectedPointsOnImagePlane1And2[CameraProjectedPointsK1_,
  CameraProjectedPointsK2_, G1_, G2_] := Module[{ImagePlaneC1Points,
                                         Modul
   ImagePlaneC2Points, SensorProjectionMtx1, SensorProjectionMtx2, PixelPitch},
  (*In this case the Image plane Point is equal to the sensor points
    eingegebenes Objekt Bild      Punkt
    which is nessecary for the derivation of the fundamental matrix.
    → Take Pitchel pitch of 1 means that only the third
      ⌊entferne
     element of the Vector is replaced by the forth*)
   PixelPitch = 100;

   ImagePlaneC1Points = Map[{#[[1]], #[[2]], #[[4]]} &, CameraProjectedPointsK1];
                        ⌊wende an
   ImagePlaneC2Points = Map[{#[[1]], #[[2]], #[[4]]} &, CameraProjectedPointsK2];
                        ⌊wende an

   Print["ImagePlaneC1Points = ", MatrixForm[Simplify[ImagePlaneC1Points]]];
   ⌊gib aus                       ⌊Matritzenform ⌊vereinfache
   Print["ImagePlaneC2Points = ", MatrixForm[N[ImagePlaneC2Points]]];
   ⌊gib aus                       ⌊Matritzenform ⌊numerischer Wert

   ComputeFundamentalMatrix[ImagePlaneC1Points, ImagePlaneC2Points, G1, G2];

  ];


(*Compute Fundamentalmatrix with 8-Point-Algorithm_______*)
                                  ⌊Punkt
ComputeFundamentalMatrix[ImagePlaneC1Points_, ImagePlaneC2Points_, G1_, G2_] :=
  Module[{CoefficientMtx, ns, F, lC1, lPrimeC1,
  ⌊Modul
    lC2, lPrimeC2, e, K1, K2, EMtx, ePrime, EpipoleLines},
   CoefficientMtx = ConstantArray[0, {9, 9}];
                    ⌊konstantes Array
   For[i = 1, i ≤ 9, i++,
   ⌊For-Schleife
    CoefficientMtx[[i]] =
      {ImagePlaneC2Points[[i, 1]] * ImagePlaneC1Points[[i, 1]],
       ImagePlaneC2Points[[i, 1]] * ImagePlaneC1Points[[i, 2]], ImagePlaneC2Points[[
        i, 1]], ImagePlaneC2Points[[i, 2]] * ImagePlaneC1Points[[i, 1]],
       ImagePlaneC2Points[[i, 2]] * ImagePlaneC1Points[[i, 2]], ImagePlaneC2Points[[
        i, 2]], ImagePlaneC1Points[[i, 1]], ImagePlaneC1Points[[i, 2]], 1};
   ];

   Print["CoefficientMtx = ", MatrixForm[N[CoefficientMtx]]];
   ⌊gib aus                   ⌊Matritzenform ⌊numerischer Wert
   Print["MatrixRank[CoefficientMtx]", MatrixRank[CoefficientMtx]];
   ⌊gib aus ⌊Rang der Matrix           ⌊Rang der Matrix
   (*Print[
     ⌊gib aus
     "RowReduce CoefficientMtx = "[MatrixForm[RowReduce[N[CoefficientMtx]]]]];*)
      ⌊reduziere Zellen            ⌊Matritzenform ⌊reduziere Ze· ⌊numerischer Wert

   Flatten[ns = NullSpace[N[CoefficientMtx]]];
   ⌊ebne ein    ⌊Nullraum  ⌊numerischer Wert
   Print["ns =", ns];
   ⌊gib aus
   F = {{ns[[1, 1]], ns[[1, 2]], ns[[1, 3]]},
     {ns[[1, 4]], ns[[1, 5]], ns[[1, 6]]}, {ns[[1, 7]], ns[[1, 8]], ns[[1, 9]]}};
   Print["F = ", MatrixForm[N[F]]];
   ⌊gib aus     ⌊Matritzenform ⌊numerischer Wert
```

```
gib aus       Matritzenform numerischer Wert

lC1 = ConstantArray[0, {9, 3}];
     konstantes Array
lPrimeC1 = ConstantArray[0, {9, 3}];
          konstantes Array

For[i = 1, i ≤ 9, i++,
For–Schleife
 lC1[[i]] = N[F.ImagePlaneC1Points[[i, All]]];
           numerischer Wert            alle
 lPrimeC1[[i]] = N[Transpose[F].ImagePlaneC2Points[[i, All]]]
                 ·· transponiere                    alle
];
Print["lC1 = ", lC1];
gib aus
Print["lPrimeC1 = ", N[lPrimeC1]];
gib aus              numerischer Wert
e = Flatten[ NullSpace[F]];
   ebne ein Nullraum
ePrime = Flatten[NullSpace[Transpose[F]]];
        ebne ein Nullraum  transponiere

Print["e = ", N[e]];
gib aus       numerischer Wert
Print["e' = ", N[ePrime]];
gib aus        numerischer Wert

EpipoleLines = Map[Cross[#, ePrime] &, ImagePlaneC2Points];
              w··· Kreuzprodukt
EpipoleLines = Map[# / #[[3]] &, EpipoleLines];
              wende an
Print["EpipoleLines = ", MatrixForm[N[EpipoleLines]]];
gib aus                  Matritzenform numerischer Wert

Print[Show[G2, ContourPlot[lC1[[1]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
gib aus zeige an Konturgraphik
  ContourPlot[lC1[[2]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
  Konturgraphik
  ContourPlot[lC1[[3]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
  Konturgraphik
  ContourPlot[lC1[[4]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
  Konturgraphik
  ContourPlot[lC1[[5]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
  Konturgraphik
  ContourPlot[lC1[[6]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
  Konturgraphik
  ContourPlot[lC1[[7]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
  Konturgraphik
  ContourPlot[lC1[[8]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
  Konturgraphik
  ContourPlot[lC1[[9]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}], G1,
  Konturgraphik
  ContourPlot[lPrimeC1[[1]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
  Konturgraphik
```

```
      Konturgraphik
      ContourPlot[lPrimeC1[[2]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
      Konturgraphik
      ContourPlot[lPrimeC1[[3]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
      Konturgraphik
      ContourPlot[lPrimeC1[[4]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
      Konturgraphik
      ContourPlot[lPrimeC1[[5]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
      Konturgraphik
      ContourPlot[lPrimeC1[[6]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
      Konturgraphik
      ContourPlot[lPrimeC1[[7]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
      Konturgraphik
      ContourPlot[lPrimeC1[[8]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
      Konturgraphik
      ContourPlot[lPrimeC1[[9]].{x, y, 1} == 0, {x, -10, 10}, {y, -5, 5}],
      Konturgraphik
      PlotRange -> All]];
      Koordinatenbe··· alle

  ComputeEssentialMtxFromFormular[F, ImagePlaneC1Points, ImagePlaneC2Points];
  Rectification[F, e, ePrime, ImagePlaneC1Points, ImagePlaneC2Points];
  NewRectification[F, e, ePrime, ImagePlaneC1Points, ImagePlaneC2Points];

  (*ComputeEssentialMtx[ImagePlaneC1Points,ImagePlaneC2Points];*)


];
```