Warping realist art to ensure consistent perspective

A new software tool for art investigations

Yu-Sung Chang
Wolfram Research, Inc.

David G. Stork
Rambus Labs

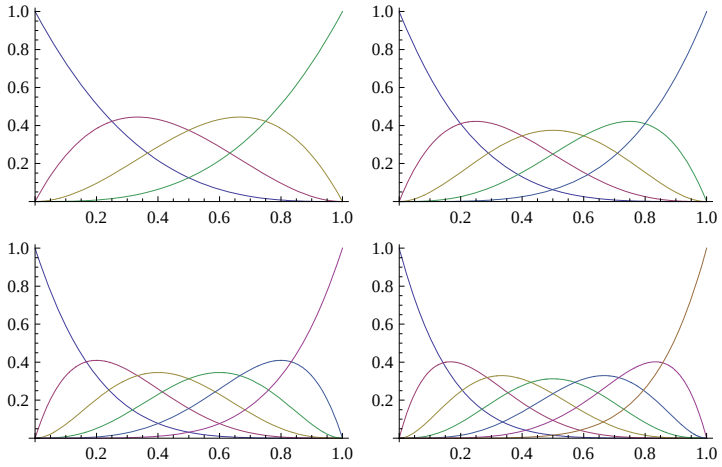
Background

Mathematica

- Powerful symbolic and numerical computation
- Spline support
- Image processing support
- CDF: Computable Document Format

B-Spline Functions

Basis functions



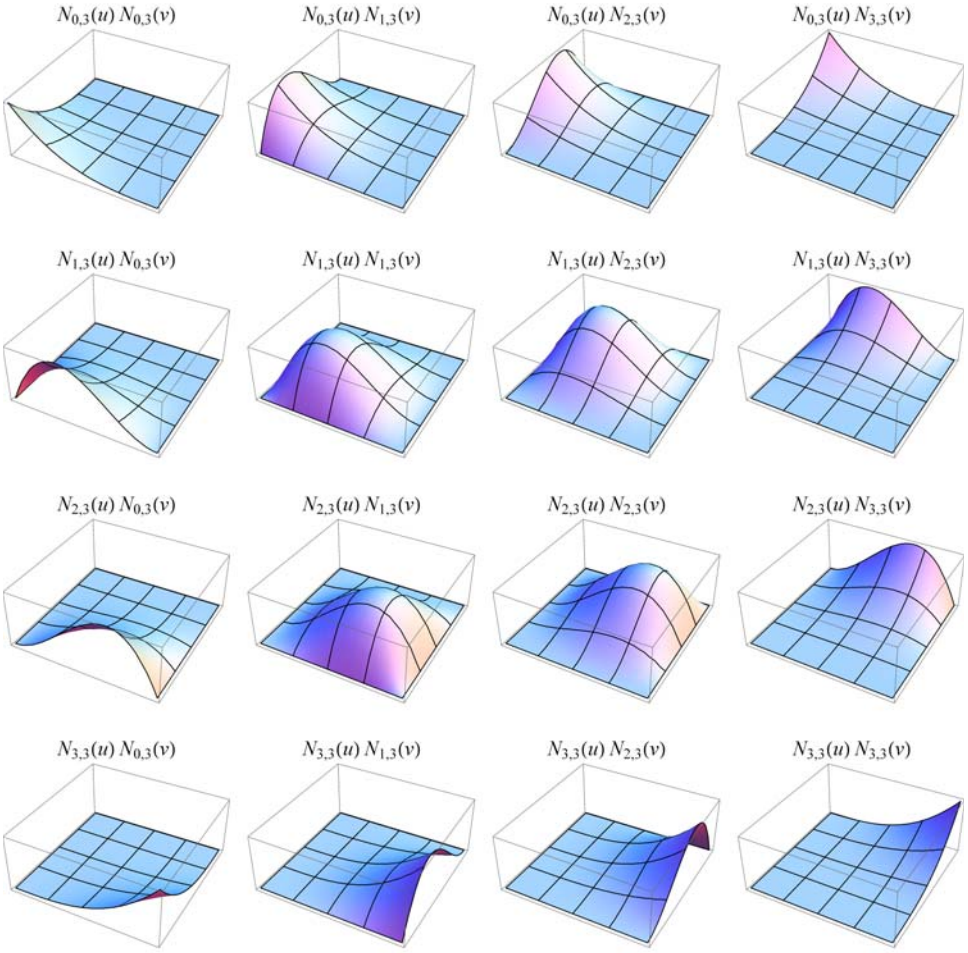
Properties

$$B_{0,3}(x) = \begin{cases} 1 & x == 0 \\ -(-1+x)^3 & 0 < x < 1 \\ 0 & \text{True} \end{cases} \qquad B_{1,3}(x) = \begin{cases} 3(-1+x)^2 x & 0 < x < 1 \\ 0 & \text{True} \end{cases}$$

$$B_{2,3}(x) = \begin{cases} -3(-1+x)x^2 & 0 < x < 1 \\ 0 & \text{True} \end{cases} \qquad B_{3,3}(x) = \begin{cases} 1 & x == 1 \\ x^3 & 0 < x < 1 \\ 0 & \text{True} \end{cases}$$

B-Spline Functions in 2D

Tensor-product of 1D basis functions

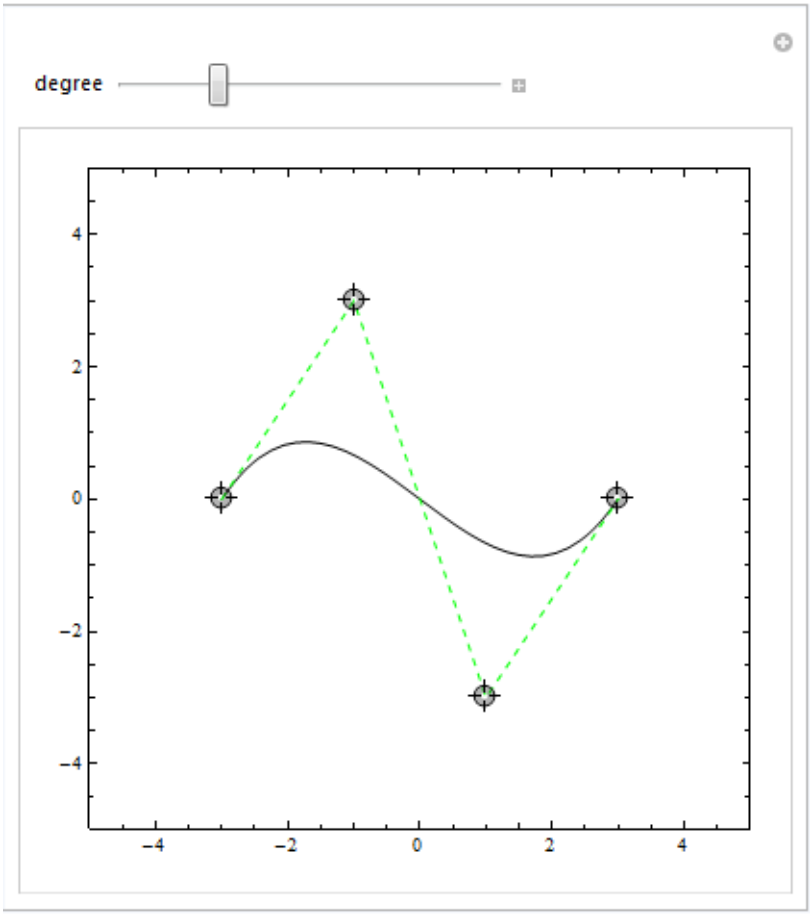


B-Spline Interpolation

In general, B-spline manifolds are not interpolating

In[4]:=
```
Manipulate[Graphics[{BezierCurve[pts, SplineDegree → d], Dashed, Green, Line[pts]},
  PlotRange → 5, Frame → True], {{pts, {{-3, 0}, {-1, 3}, {1, -3}, {3, 0}}},
  Locator, LocatorAutoCreate → True}, {{d, 3, "degree"}, 2, 6, 1}]
```

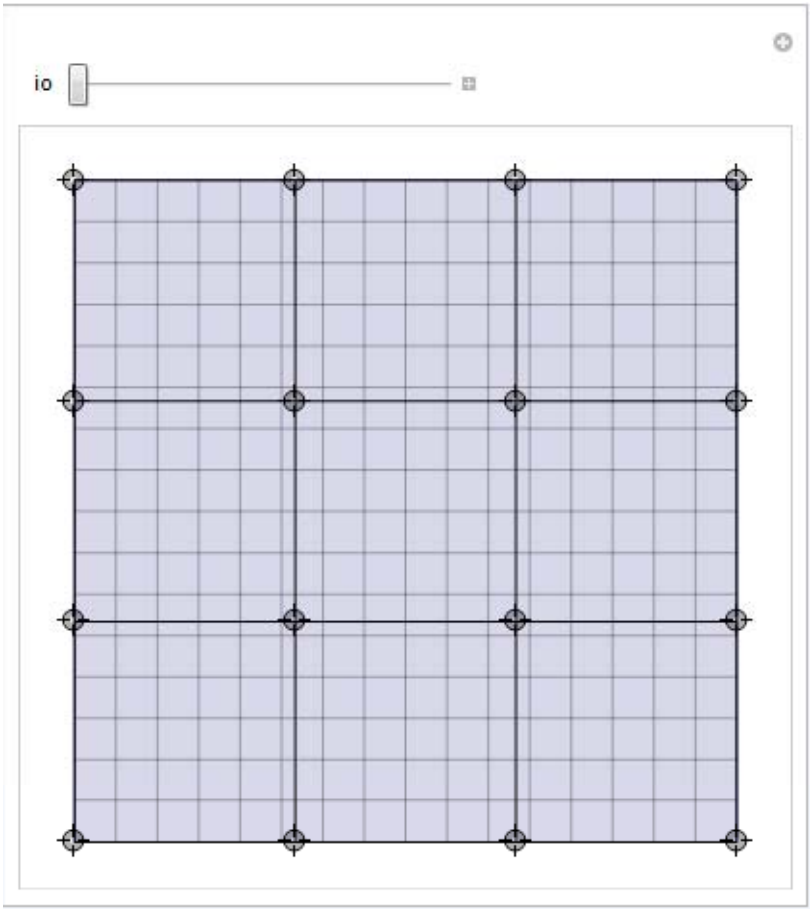
Out[4]=



Interpolation of 2D B-spline surface

In[5]:=
```
With[{nx = 4, ny = 4}, Manipulate[
  sourcePts = Flatten[Table[{i, j}, {i, 0, 1, 1 / (nx - 1)}, {j, 0, 1, 1 / (ny - 1)}], 1];
  f = Interpolation[MapThread[{#1, #2} &,
     {Flatten[Partition[sourcePts, ny], 1], Flatten[Partition[targetPts, ny], 1]}],
    Method → "Spline", InterpolationOrder → io];
  Graphics[{ParametricPlot[f[u, v], {u, 0, 1}, {v, 0, 1}][[1]],
    Line /@ Partition[targetPts, nx], Line /@ Transpose[Partition[targetPts, nx]]}],
 {{targetPts, Flatten[Table[{i, j}, {i, 0, 1, 1 / (nx - 1)}, {j, 0, 1, 1 / (ny - 1)}], 1]},
  Locator}, {io, 1, 3, 1}]]
```

Out[5]=



Using Texture on 2D B-spline interpolation surface

In[6]:=
```
With[{nx = 4, ny = 4}, Manipulate[
  sourcePts = Flatten[Table[{i, j}, {i, 0, 1, 1 / (nx - 1)}, {j, 0, 1, 1 / (ny - 1)}], 1];
  f = Interpolation[MapThread[{#1, #2} &,
     {Flatten[Partition[sourcePts, ny], 1], Flatten[Partition[targetPts, ny], 1]}],
    Method → "Spline", InterpolationOrder → io];
  Graphics[{Dynamic@Texture[CurrentImage[]],
    ParametricPlot[f[u, v], {u, 0, 1}, {v, 0, 1}, Mesh → None,
      PlotStyle → Opacity[1], TextureCoordinateFunction → ({#3, #4} &)][[1]],
    Line /@ Partition[targetPts, nx], Line /@ Transpose[Partition[targetPts, nx]]}],
  {{targetPts, Flatten[Table[{i, j}, {i, 0, 1, 1 / (nx - 1)}, {j, 0, 1, 1 / (ny - 1)}], 1]},
   Locator}, {io, 1, 3, 1}]]
```

Implementation

Code

In[1]:=
```
Clear[imageWarping];

imageWarping[img_, setup_: Automatic] :=
 DynamicModule[{mode = 1, io = 1, nx = 6,
   ny = 4, pnx = 6, pny = 4, selIdx = 0, tConst = True, tIdx = False,
   tLoc = True, tGrids = False, tPlines = False, tPlinesMono = False,
   tPguides = False, tPcurve = False, grids = {}, trans = 1,
   sourcePts = Flatten[Table[{i, j}, {i, 0, 1, 1 / 5(* nx-1 *)},
      {j, 0, 1, 1 / 3(* ny-1 *)}], 1],
   targetPts = Flatten[Table[{i, j}, {i, 0, 1, 1 / 5(* nx-1 *)},
      {j, 0, 1, 1 / 3(* ny-1 *)}], 1],
   g, f, pLines = {}, pMode = 1, pStack = {}, pI, uStart = 0,
   uEnd = 1, cRendering = Automatic,
   imgSize = ImageDimensions[img][[1]],
   imgPadLeft = 10, imgPadRight = 10, imgPadBottom = 10, imgPadTop = 10},

  If[setup =!= Automatic,
   If[Quiet@OptionValue[setup, "GridDimensions"] =!= "GridDimensions",
    {nx, ny} = {pnx, pny} = OptionValue[setup, "GridDimensions"]];
   If[Quiet@OptionValue[setup, "InterpolationOrder"] =!= "InterpolationOrder",
    io = OptionValue[setup, "InterpolationOrder"]];
   If[Quiet@OptionValue[setup, "SourceControlPoints"] =!= "SourceControlPoints",
    sourcePts = OptionValue[setup, "SourceControlPoints"]];
   If[Quiet@OptionValue[setup, "TargetControlPoints"] =!= "TargetControlPoints",
    targetPts = OptionValue[setup, "TargetControlPoints"]];
   If[Quiet@OptionValue[setup, "ParallelLines"] =!= "ParallelLines",
    pLines = OptionValue[setup, "ParallelLines"]];
  ];

  Style[
   Pane[
    Grid[{{

       Dynamic[
        If[pnx =!= nx || pny =!= ny, sourcePts =
          Flatten[Table[{i, j}, {i, 0, 1, 1 / (nx - 1)}, {j, 0, 1, 1 / (ny - 1)}], 1];
         pnx = nx; targetPts = Flatten[Table[{i, j}, {i, 0, 1, 1 / (nx - 1)},
            {j, 0, 1, 1 / (ny - 1)}], 1]; pny = ny];
        f = Interpolation[MapThread[{#1, #2} &, {Flatten[
              padding[Partition[sourcePts, ny], io], 1], Flatten[padding[
```

```
      Partition[(1 - trans) sourcePts + trans targetPts, ny], io], 1]}],
  Method → "Spline", InterpolationOrder → io];
With[{dom = f["Domain"]}, g[x_, y_] := Piecewise[{{f[x, y], dom[[1, 1]] ≤
      x ≤ dom[[1, 2]] && dom[[2, 1]] ≤ y ≤ dom[[2, 2]]}, {{x, y}, True}}]];

Panel[
 Switch[mode,
  1,
  EventHandler[
   LocatorPane[
    Dynamic[sourcePts, (
       With[
        {diff =
          Which[
           CurrentValue["ShiftKey"],
           Function[{pt}, {1., 0.} pt] /@ (sourcePts - #),
           CurrentValue["ControlKey"], Function[{pt},
             {0., 1.} pt] /@ (sourcePts - #),
           True, sourcePts - #
          ]},
        With[{newPts = sourcePts - diff},
         selIdx =
          Quiet@First@First@Position[diff, {x_, y_} /; x ≠ 0 || y ≠ 0];
         sourcePts = targetPts = If[selIdx === {},
            newPts,
            Table[
             Which[
              Floor[(i - 1) / ny] == Floor[(selIdx - 1) / ny],
              {newPts[[selIdx, 1]], newPts[[i, 2]]},
              Mod[i - 1, ny] == Mod[selIdx - 1, ny], {newPts[[
                i, 1]], newPts[[selIdx, 2]]},
              True, newPts[[i]]
             ], {i, nx ny}]]
        ]
       ]) &],

    Dynamic[
     Graphics[{Texture[img], Polygon[{{0, 0}, {1, 0}, {1, 1}, {0, 1}},
        VertexTextureCoordinates → {{0, 0}, {1, 0}, {1, 1}, {0, 1}}],
       If[tGrids, {
         White, Opacity[.7], AbsoluteThickness[3],
         Line[
          Table[{sourcePts[[1 + ny (i - 1)]], sourcePts[[ny i]]}, {i, nx}]],
         Line[Table[{sourcePts[[i]], sourcePts[[
             i + ny (nx - 1)]]}, {i, ny}]],
         Black, AbsoluteThickness[2],
         Line[
          Table[{sourcePts[[1 + ny (i - 1)]], sourcePts[[ny i]]}, {i, nx}]],
```

```
      Line[Table[{sourcePts[[i]], sourcePts[[
          i + ny (nx - 1)]]}, {i, ny}]]
     }, {}],
    If[tPlines,
     If[tPlinesMono,
      {
       White, Opacity[.5], AbsoluteThickness[3],
       If[pLines =!= {}, {Line[(stretch[#[[1]], #[[2]]] &) /@ pLines],
         AbsolutePointSize[8], Point /@ pLines}, {}],
       ColorData["HTML", "DarkBlue"], Opacity[.5],
       AbsoluteThickness[2],
       If[pLines =!= {}, {AbsolutePointSize[5], MapIndexed[{Line[
             stretch[#1[[1]], #1[[2]]]], Point[#1]} &, pLines]}, {}]
      },
      {
       Black, Opacity[.7], AbsoluteThickness[5],
       If[pLines =!= {}, {Line[(stretch[#[[1]], #[[2]]] &) /@ pLines],
         AbsolutePointSize[11], Point /@ pLines}, {}],
       Yellow, Opacity[1], AbsoluteThickness[2],
       If[pLines =!= {}, {AbsolutePointSize[8],
         MapIndexed[{ColorData[24][3 Ceiling[First[#2] / 2] - 1], Line[
             stretch[#1[[1]], #1[[2]]]], Point[#1]} &, pLines]}, {}]
      }],
     {}],
    If[tIdx, MapIndexed[{Opacity[.7], Text[Style[First[#2],
          FontFamily → "Arial", 16, Bold, Black], Offset[{22, -2}, #1]],
       Opacity[1], Text[Style[First[#2], FontFamily → "Arial",
         16, Bold, Yellow], Offset[{20, 0}, #1]]} &, sourcePts], {}],
    If[pStack =!= {}, {Black, Opacity[.7], AbsolutePointSize[5],
      Point[pStack], Red, Opacity[1],
      AbsolutePointSize[3], Point[pStack]}, {}]

   }, FrameTicks → None, Frame → True, FrameStyle → GrayLevel[.7],
   PlotRangePadding → 0.001, PlotRange → {{0, 1}, {0, 1}},
   PlotRangeClipping → True, AspectRatio → ImageAspectRatio[img],
   ImageSize → Dynamic[imgSize], ImagePadding → Dynamic[
     {{imgPadLeft, imgPadRight}, {imgPadBottom, imgPadTop}}]],
  TrackedSymbols :↦ {sourcePts, targetPts, tGrids, tPlines,
    tPlinesMono, pLines, pStack, tIdx}],

 Appearance → Graphics[Dynamic[If[tLoc,
     {White, AbsoluteThickness[3], Opacity[.7],
      Circle[{0, 0}, Offset[{6, 6}]], Black, Opacity[1],
      AbsoluteThickness[1], Circle[{0, 0}, Offset[{6, 6}]]},
     {}], TrackedSymbols :↦ {tLoc}]]
]
, {{"MouseDown", 2} :↦ (
   Switch[pMode,
```

```
      1, Beep[]; If[pStack === {},
       pStack = MousePosition["Graphics"], AppendTo[pLines,
        {pStack, MousePosition["Graphics"]}]; pStack = {}],
      -1, pI = 1; If[pLines =!= {},
       With[{pt = MousePosition["Graphics"]}, While[(Length[pLines] ≥
             pI) && (EuclideanDistance[pLines[[pI, 1]], pt] > 0.01) &&
          (EuclideanDistance[pLines[[pI, 2]], pt] > 0.01), pI++]];
       If[pI ≤ Length[pLines], Beep[]; pLines = Delete[pLines, pI]]]
     ]
    )}],

2,
LocatorPane[
 Dynamic[targetPts, (
    With[
     {diff =
       Which[
        CurrentValue["ShiftKey"],
        Function[{pt}, {1., 0.} pt] /@ (targetPts - #),
        CurrentValue["ControlKey"], Function[{pt},
          {0., 1.} pt] /@ (targetPts - #),
        True, targetPts - #
       ]},
     With[{newPts = targetPts - diff},
      If[tConst,
       selIdx =
        Quiet@First@First@Position[diff, {x_, y_} /; x ≠ 0 || y ≠ 0];
       targetPts = If[selIdx === {},
         #,
         Table[
          Which[
           Floor[(i - 1) / ny] == Floor[(selIdx - 1) / ny],
           {newPts[[selIdx, 1]], newPts[[i, 2]]},
           Mod[i - 1, ny] == Mod[selIdx - 1, ny], {newPts[[
             i, 1]], newPts[[selIdx, 2]]},
           True, newPts[[i]]
          ], {i, nx ny}]],
       targetPts = newPts]
     ]
    ) &],

 Dynamic[
  ParametricPlot[f[x, y], {x, 0, 1}, {y, 0, 1}, BoundaryStyle → None,
   Frame → True, FrameTicks → None, FrameStyle → GrayLevel[.7],
   TextureCoordinateFunction → ({#3, #4} &), MaxRecursion → 4,
   PlotStyle → Directive[Opacity[1], Texture[img]], Mesh → None,
   PlotRange → {{0, 1}, {0, 1}}, PlotRangePadding → 0.001,
   AspectRatio → ImageAspectRatio[img], ImageSize → Dynamic[imgSize],
```

```
ImagePadding → Dynamic[{{imgPadLeft, imgPadRight},
   {imgPadBottom, imgPadTop}}], Frame → False, Axes → False,
Epilog → {
  Green, AbsoluteThickness[1],
  With[{pts = (1 - trans) sourcePts + trans targetPts}, {
    If[tGrids, {
      White, Opacity[.7], AbsoluteThickness[3],
      Table[Line[Table[{pts[[1 + j + ny (i - 1)]],
          pts[[2 + j + ny (i - 1)]]}, {i, nx}]], {j, 0, ny - 2}],
      Table[Line[Table[{pts[[i + ny (j - 1)]], pts[[i + ny j]]},
         {i, ny}]], {j, 1, nx - 1}],
      Black, AbsoluteThickness[2],
      Table[Line[Table[{pts[[1 + j + ny (i - 1)]],
          pts[[2 + j + ny (i - 1)]]}, {i, nx}]], {j, 0, ny - 2}],
      Table[Line[Table[{pts[[i + ny (j - 1)]], pts[[i + ny j]]},
         {i, ny}]], {j, 1, nx - 1}]
     }, {}],
    If[tPlines && ! tPcurve,
     If[tPlinesMono,
      {
       White, Opacity[.5],
       AbsoluteThickness[3], AbsolutePointSize[8],
       If[pLines =!= {}, {Line[stretch[f @@ #[[1]], f @@ #[[2]]] & /@
           pLines], Point[f @@@ Flatten[pLines, 1]]}, {}],
       ColorData["HTML", "DarkBlue"], Opacity[.5],
       AbsoluteThickness[2], AbsolutePointSize[5],
       If[pLines =!= {}, MapIndexed[{Line[stretch[f @@ #1[[1]],
            f @@ #1[[2]]]], Point[f @@@ #1]} &, pLines], {}]
      },
      {
       Black, Opacity[.7],
       AbsoluteThickness[5], AbsolutePointSize[11],
       If[pLines =!= {}, {Line[stretch[f @@ #[[1]], f @@ #[[2]]] & /@
           pLines], Point[f @@@ Flatten[pLines, 1]]}, {}],
       Yellow, Opacity[1], AbsoluteThickness[2],
       AbsolutePointSize[8],
       If[pLines =!= {}, MapIndexed[{ColorData[24][
            3 Ceiling[First[#2] / 2] - 1], Line[stretch[f @@ #1[[1]],
            f @@ #1[[2]]]], Point[f @@@ #1]} &, pLines], {}]
      }],
     {}],
    If[tPlines && tPcurve,
     If[tPlinesMono,
      {
       White, Opacity[.5],
       AbsoluteThickness[3], AbsolutePointSize[8],
       If[pLines =!= {}, {curvelinear[f, #, uStart, uEnd, Black] & /@
          pLines, Point[f @@@ Flatten[pLines, 1]]}, {}],
```

```
                ColorData["HTML", "DarkBlue"], Opacity[.5],
                AbsoluteThickness[2], AbsolutePointSize[5],
                If[pLines =!= {}, {curvelinear[f, #, uStart, uEnd,
                     ColorData["HTML", "DarkBlue"]] & /@
                   pLines, Point[f @@@ Flatten[pLines, 1]]}, {}]
               },
               {
                Black, Opacity[.7],
                AbsoluteThickness[5], AbsolutePointSize[11],
                If[pLines =!= {}, {curvelinear[f, #, uStart, uEnd, Black] & /@
                   pLines, Point[f @@@ Flatten[pLines, 1]]}, {}],
                Yellow, Opacity[1], AbsoluteThickness[2],
                AbsolutePointSize[8],
                If[pLines =!= {}, MapIndexed[{curvelinear[f, #1, uStart,
                     uEnd, ColorData[24][3 Ceiling[First[#2] / 2] - 1]],
                    ColorData[24][3 Ceiling[First[#2] / 2] - 1],
                    Point[f @@@ #1]} &, pLines], {}]
               }],
              {}]
            }]
          }
       ], TrackedSymbols :>
        {f, tGrids, tPlines, tPlinesMono, tPcurve, uStart, uEnd, trans}],

     Appearance → Graphics[Dynamic[If[tLoc,
          (*{Black,Opacity[.5],AbsolutePointSize[13],Point[{0,0}],Opacity[1],
           LightBlue,AbsolutePointSize[10],Point[{0,0}]}*)
          {White, AbsoluteThickness[3], Opacity[.7],
           Circle[{0, 0}, Offset[{6, 6}]], Black, Opacity[1],
           AbsoluteThickness[1], Circle[{0, 0}, Offset[{6, 6}]]},
          {}], TrackedSymbols :> {tLoc}]]
    ]
   ]
  ],
  TrackedSymbols :> {mode, trans, io, nx, ny, sourcePts, targetPts}],

 Panel@Column[{
    "control points:",
    RadioButtonBar[Dynamic[mode], {1 → "source", 2 → "target"}],
    "",
    "interpolation order:",
    RadioButtonBar[Dynamic[io], {1, 2, 3}],
    "",
    "number of grid points",
    Row[{"x: ",
      Slider[Dynamic[nx], {2, 10, 1}, ImageSize → Small], " ", Dynamic[nx]}],
    Row[{"y: ", Slider[Dynamic[ny], {2, 10, 1}, ImageSize → Small],
      " ", Dynamic[ny]}],
```

```
    "",
    Grid[{
      {"constrained control:", Checkbox[Dynamic[tConst]]},
      {"", ""},
      {"show locators:", Checkbox[Dynamic[tLoc]]},
      {"show grids:", Checkbox[Dynamic[tGrids]]},
      {"show parallel lines:", Checkbox[Dynamic[tPlines]]},
      {" (use curvelinear)", Checkbox[Dynamic[tPcurve]]},
      {" (use simple color)", Checkbox[Dynamic[tPlinesMono]]}
     }, Alignment → {{Left, Right}, Center}],
    "",
    Row[{"image width: ", Dynamic[imgSize]}],
    Slider[Dynamic[imgSize], {.5 ImageDimensions[img][[1]],
      2. ImageDimensions[img][[1]]}, ImageSize → {120, 20}],
    "",
    "right-click action",
    Row[{RadioButton[Dynamic[pMode, (pStack = {}; pMode = #) &], 1],
      "  add parallel lines"}],
    Row[{RadioButton[Dynamic[pMode, (pStack = {}; pMode = #) &], -1],
      "  delete parallel lines"}],
    "",
    Button["snapshot to clipboard", CopyToClipboard[{"GridDimensions" → {nx,
        ny}, "InterpolationOrder" → io, "SourceControlPoints" → sourcePts,
      "TargetControlPoints" → targetPts, "ParallelLines" → pLines}]],
    "",
    Row[{"transition: ", Dynamic[trans]}],
    Slider[Dynamic[trans], {0, 1}, ImageSize → {120, 20}],
    Animator[Dynamic[trans],
     {0, 1}, AnimationRunning → False, DisplayAllSteps → True,
     AnimationDirection → ForwardBackward, AppearanceElements →
      {"StepLeftButton", "StepRightButton", "PlayPauseButton",
       "FasterSlowerButtons", "DirectionButton", "ResetButton"}],
    "",
    "",
    OpenerView[{Style["debug", Gray],
      Column[{
        "",
        "curvelinear lines range:",
        Slider[Dynamic[uStart], {0, 1}, ImageSize → Small],
        Slider[Dynamic[uEnd], {0, 1}, ImageSize → Small],
        "",
        "control style",
        RadioButtonBar[Dynamic[cRendering], {Automatic, "Generic"}]
       }]
     }]
   }]

}}], FrameMargins → 20
```

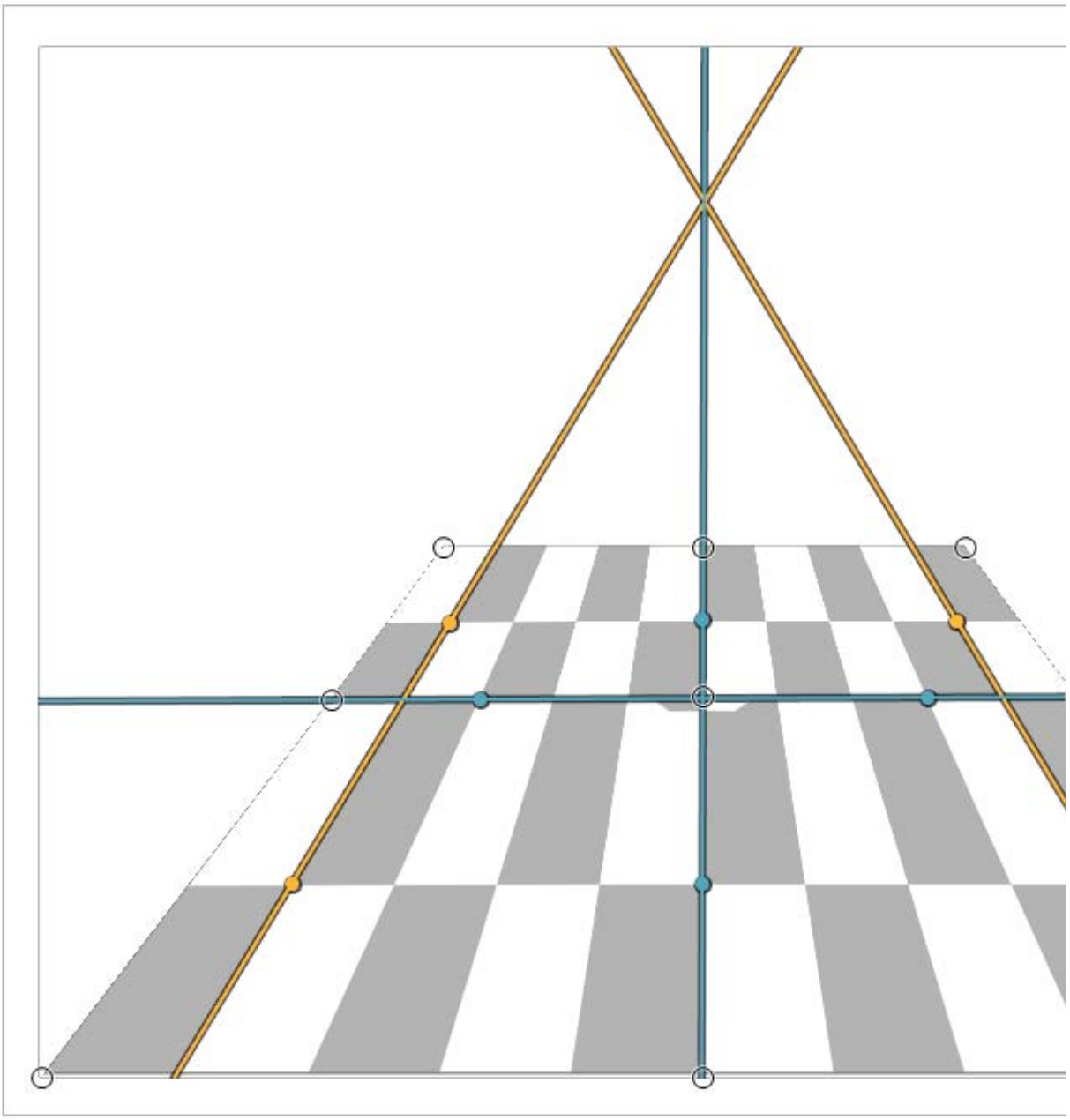
```
 ],
 ControlsRendering → Dynamic[cRendering]
], BaseStyle → "Label", Initialization :> (

  stretch[pt1_, pt2_] :=
   With[{x1 = pt1[[1]], x2 = pt2[[1]], y1 = pt1[[2]], y2 = pt2[[2]]},
    If[y1 == y2,
     {{0, (-x2 y1 + x1 y2)/(x1 - x2)}, {1, (y1 - x2 y1 - y2 + x1 y2)/(x1 - x2)}},
     {{(x2 y1 - x1 y2)/(y1 - y2), 0}, {(x1 - x2 + x2 y1 - x1 y2)/(y1 - y2), 1}}
    ]];
  curvelinear[f_, {{x1_, y1_}, {x2_, y2_}}, ustart_, uend_, style_] :=
   If[x1 == x2 || Abs[(y2 - y1)/(x2 - x1)] > 2,
    ParametricPlot[f[(x2 - x1)/(y2 - y1) (v - y2) + x2, v],
      {v, ustart, uend}, PlotStyle → style][[1, 1]],
    ParametricPlot[f[u, (y2 - y1)/(x2 - x1) (u - x2) + y2],
      {u, ustart, uend}, PlotStyle → style][[1, 1]]
   ];
  padding[pts_, n_] := pts;
)]
```

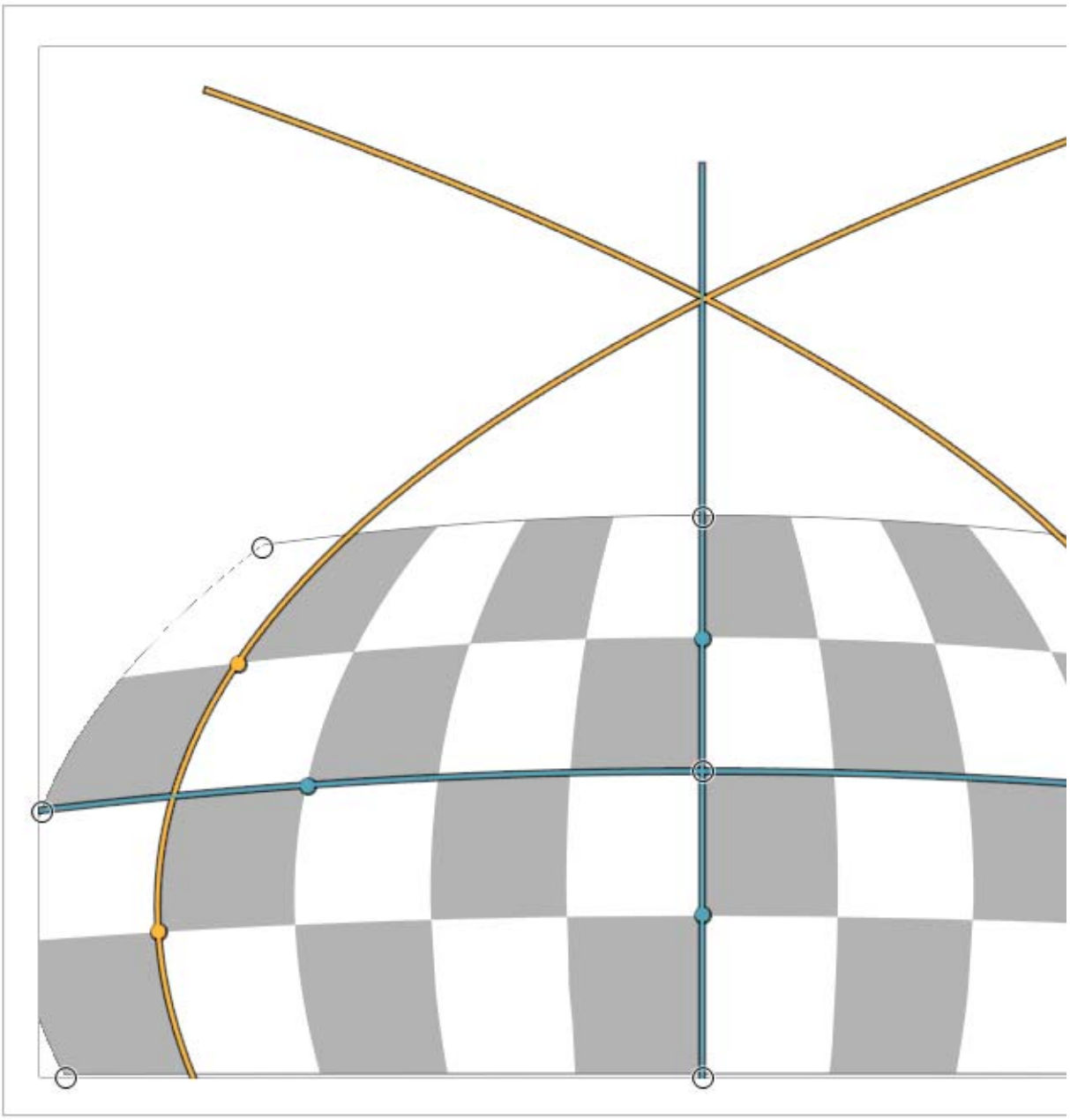
Test Pattern

In[3]:=
```
checker = ImagePad[Rasterize[Framed[
    Graphics[{EdgeForm[], Table[{If[Xor[EvenQ[i], EvenQ[j]], White, GrayLevel[.7]],
        Rectangle[{10 i, 10 j}, {10 (i + 1), 10 (j + 1)}]}, {i, 10}, {j, 4}]},
     PlotRange → All, ImageSize → 800, PlotRangePadding → 0, ImagePadding → 0],
    FrameMargins → 0, FrameStyle → Gray], "Image"], {{0, 0}, {0, 300}}}, White]
```

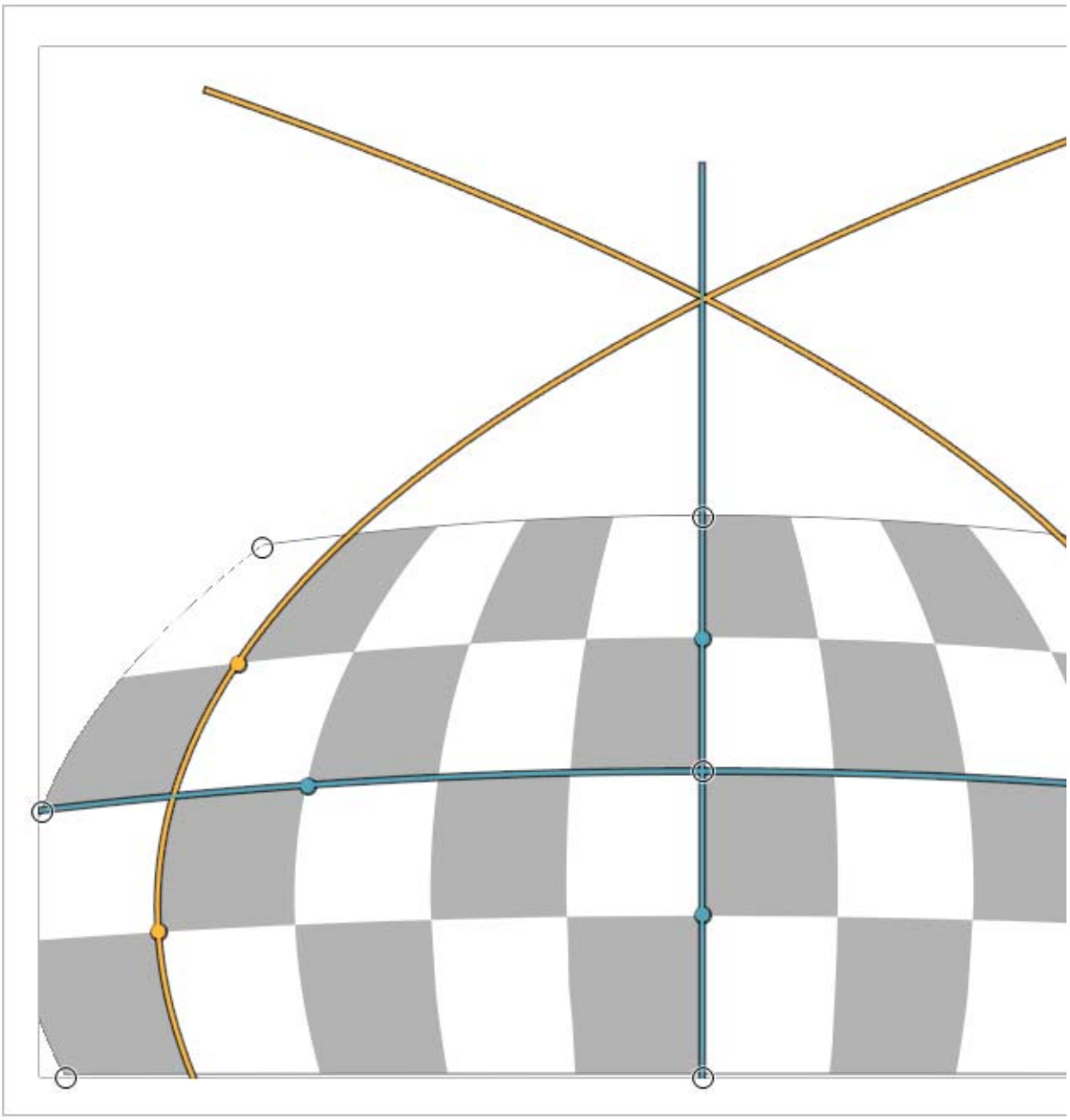
B-Spline Warping—Linear



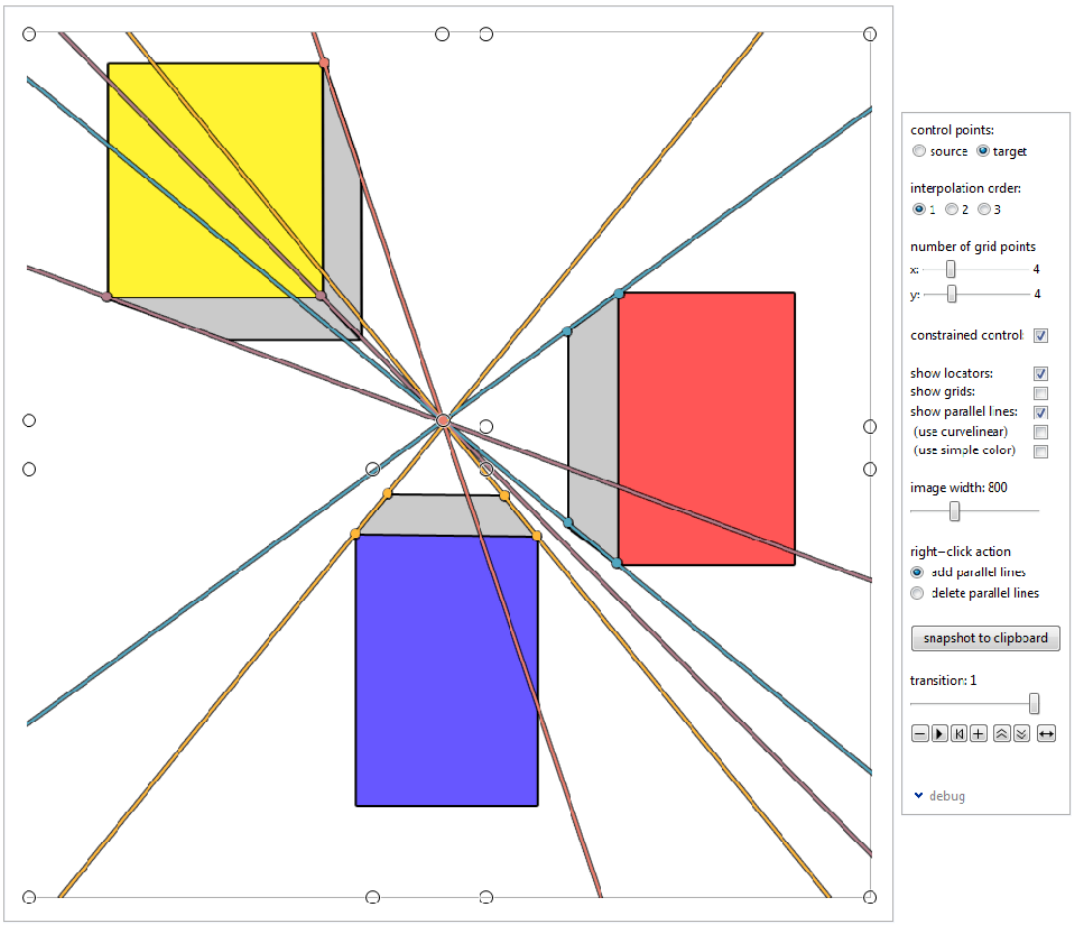
B-Spline Warping—Quadratic



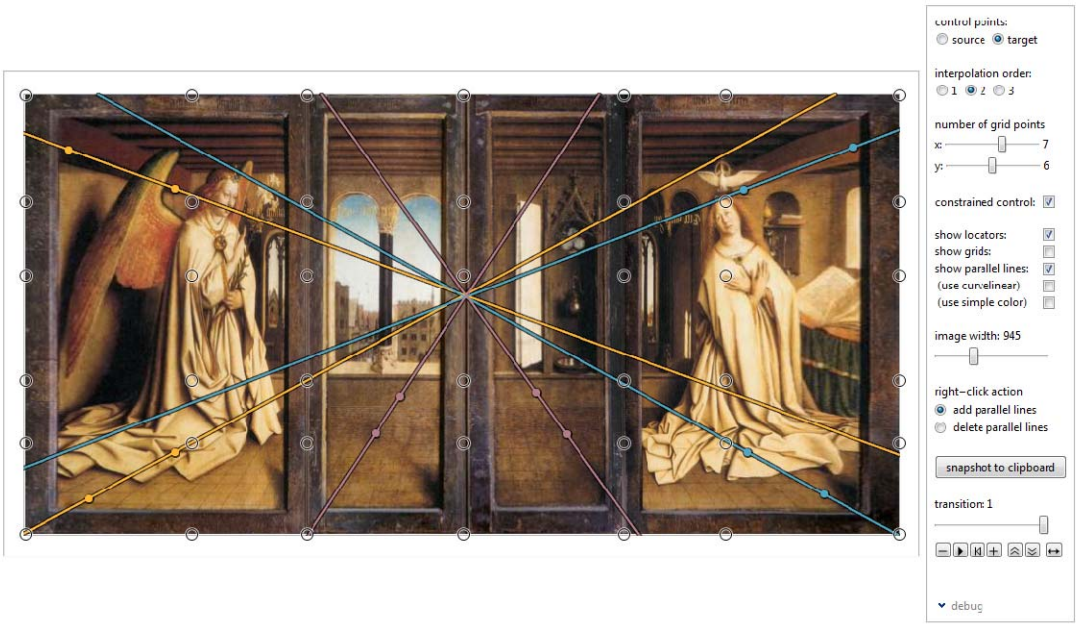
B-Spline Warping—Cubic



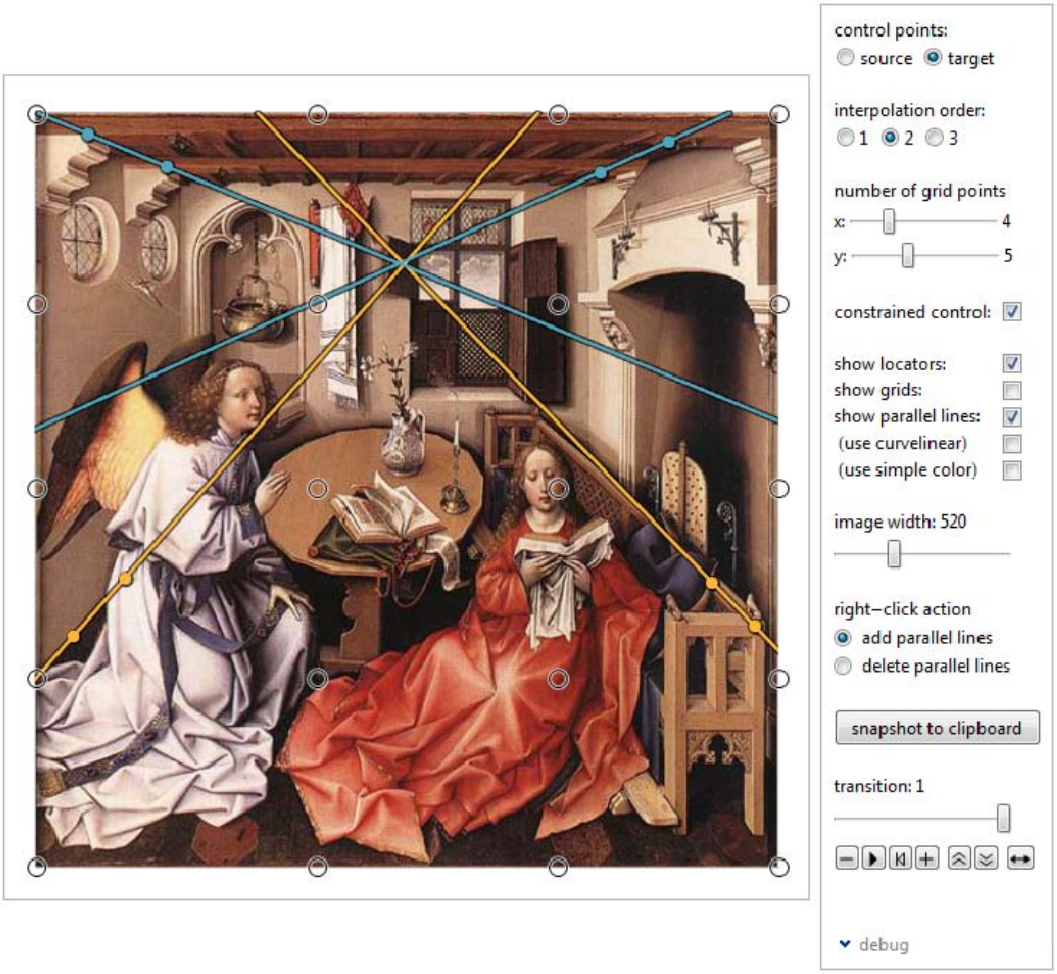
Single Vanishing Point Fixing



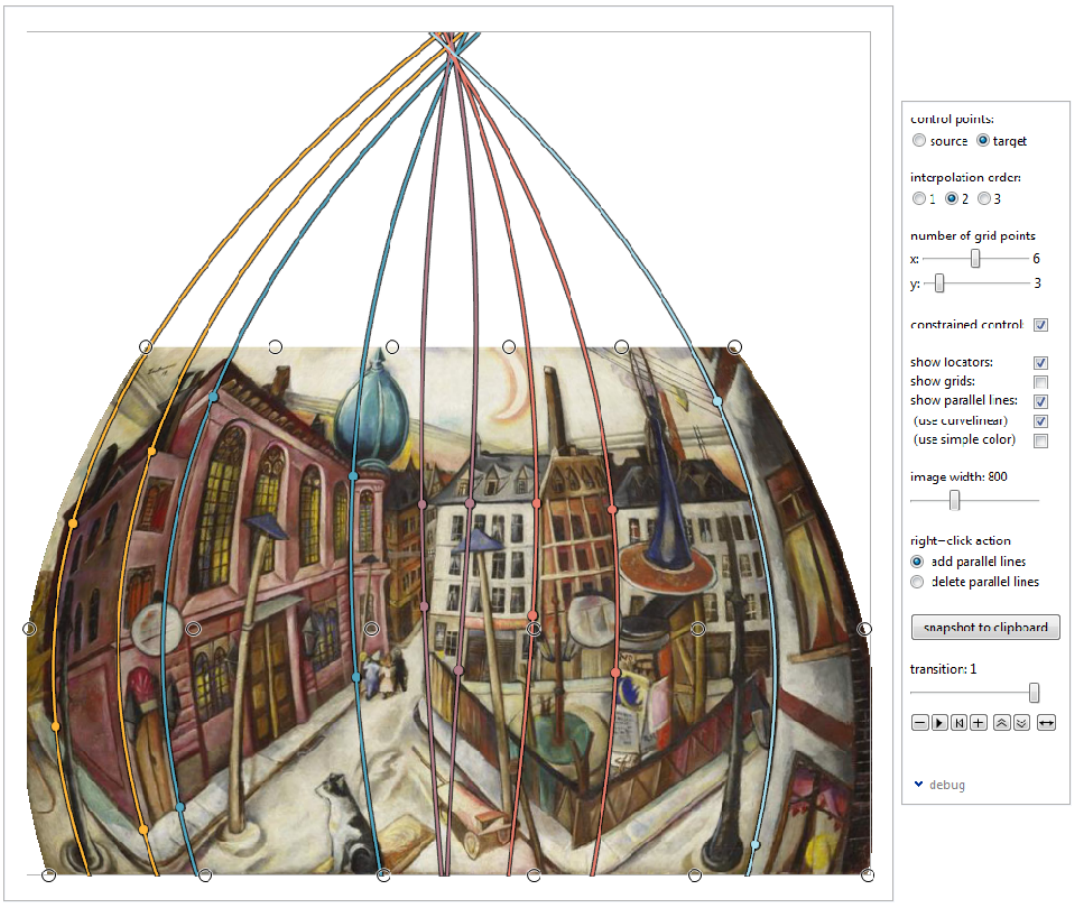
Ghent Altarpiece



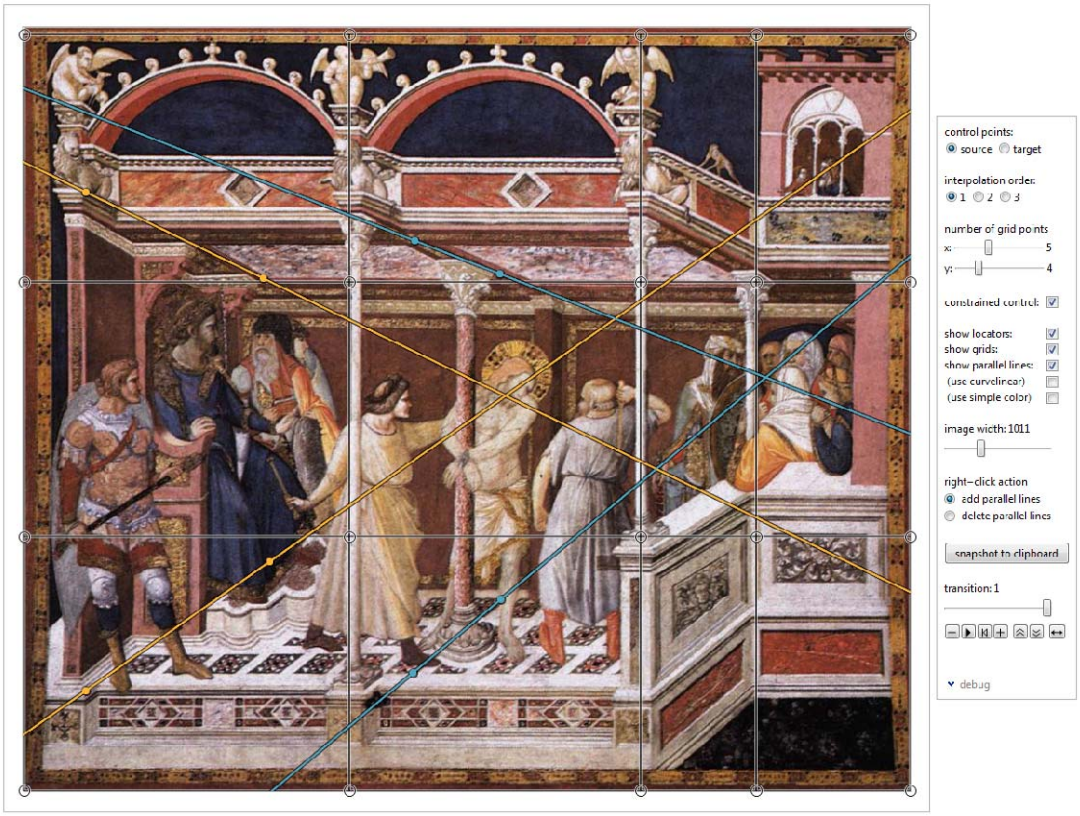
Robert Campin Merode, Altarpiece, c. 1425. Triptych



Max Beckmann, Die Synagoge in Frankfurt am Main, 1919



Pietro Lorenzetti, Flagellation of Christ



Future Work

Interactivity

Direct control on the surface (Hong *et al.*, 2000)

Touch interface: iPad, etc.

Constraints

Area preservation

Fixed points

Layering / 3D

Content-aware resizing / filling (Barnes *et al.*, 2009)

Color space management

RGB space is not linear color space: Application of B-spline interpolation method may cause distortion.

XYZ space in the future