(*    *)

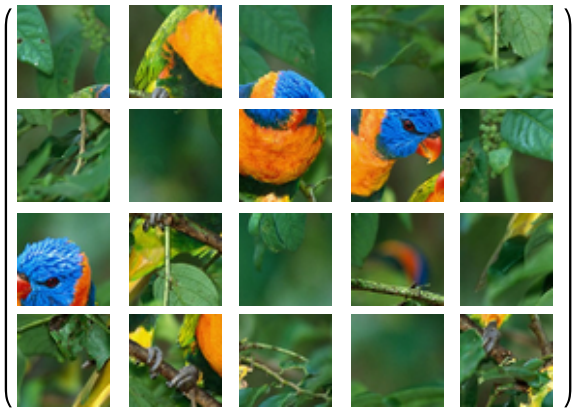
---

# Importing Image (get shuffled blocks, represented by RGB matrix)

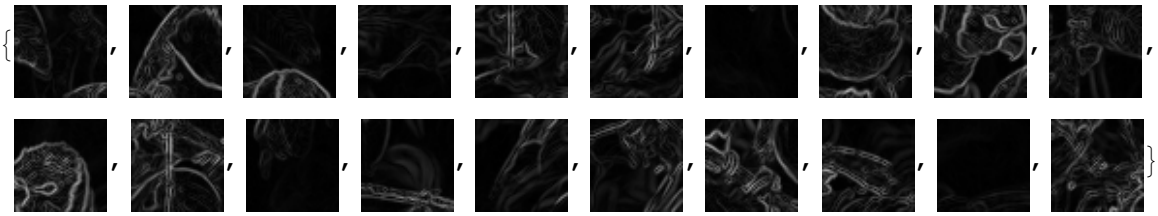
```
{partitionW, partitionH} = {5, 4};
```

im = ImageResize[  , {48 * partitionW, 48 * partitionH}];

```
{imW, imH} = ImageDimensions[im];
pxl = 48;
orig = MatrixForm[imgBlock = ImagePartition[im, pxl] ];
shuffleList = RandomSample[Range[20]];
shuf = MatrixForm[imgShuffle =
   ArrayReshape[Flatten[imgBlock][[#]] & /@ shuffleList, {imH / pxl, imW / pxl}]]
block = Flatten[imgShuffle];
```

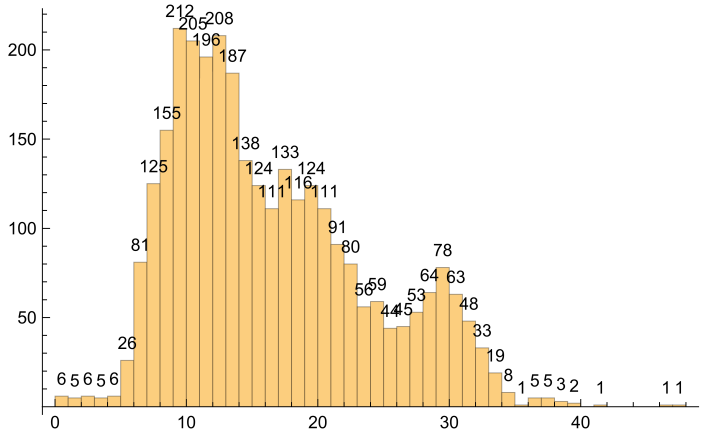


```
Table[GradientFilter[block[[i]], 1], {i, Length[block]}]
(*play with different image filters to get
 different border information for better pairing*)
```



# Define matching score and find connecting between blocks

```
{blockW, blockH} = ImageDimensions[block[[1]]];(*blockW,blockH*)
bIdx = {Table[{1, r}, {r, blockW}], Table[{r, blockW}, {r, blockH}],
   Table[{blockH, r}, {r, blockW, 1, -1}], Table[{r, 1}, {r, blockH, 1, -1}]};
SideRGB[idx_, side_] := Table[ImgD = ImageData[block[[idx]]];
   (*idx th block, side th border*)
   (ImgD[[#1, #2]] &) @@ bIdx[[side, i]], {i, Length[bIdx[[side]]]}];
compareList = Flatten[Table[{i * 4 + is, j * 4 + js},
    {i, 0, Length[block] - 2}, {is, 4}, {j, i + 1, Length[block] - 1}, {js, 4}], 3];
cDist[l1_, l2_] := ((*distance between two boarders*)
  dim = Dimensions[l1]; (*dimension of list1*)
  dim2 = Dimensions[l2]; (*dimension of list2*)
  If[dim ≠ dim2, Return["Wrong size!"]];
  (*if lists have different dimensions*)
  anti = Total[Table[EuclideanDistance[l1[[i]], Reverse[l2][[i]]], {i, dim[[1]]}]]
 )
cDiffDist[l1_, l2_] :=
 ((*experiment on other methods to define distance between two boarders. try
   different filters on original image, like edge detection and gradients*)
  dim = Dimensions[l1]; (*dimension of list1*)
  dim2 = Dimensions[l2]; (*dimension of list2*)
  If[dim ≠ dim2, Return["Wrong size!"]];
  (*if lists have different dimensions*)
  l1Diff = Differences[l1];
  l2Diff = Differences[Reverse[l2]];
  anti =
   Total[Table[EuclideanDistance[l1Diff[[i]], l2Diff[[i]]], {i, dim[[1]] - 1}]]
 )
borders = Flatten[Table[SideRGB[i, s], {i, Length[block]}, {s, 4}], 1];
Dimensions[borders];
checkByIdxSide[i1_, s1_, i2_, s2_] :=
 (*a debugging function to show the information of two boarders*)
 Row[{{i1, s1, i2, s2, cDist[SideRGB[i1, s1], SideRGB[i2, s2]]},
   MatrixPlot[{RGBColor @@@ SideRGB[i1, s1], RGBColor @@@ Reverse[SideRGB[i2, s2]]},
    FrameTicks → None, ImageSize → 200]}]
checkByBorderIdx[i1_, i2_] := (*a debugging function to show the
  information of two boarders*)Row[{{Ceiling[i1 / 4], Mod[i1, 4, 1],
    Ceiling[i2 / 4], Mod[i2, 4, 1], cDist[borders[[i1]], borders[[i2]]]},
   MatrixPlot[{RGBColor @@@ borders[[i1]], RGBColor @@@ Reverse[borders[[i2]]]},
    FrameTicks → None, ImageSize → 200]}]
Histogram[Flatten[Table[cDist[borders[[compareList[[i, 1]]]],
    borders[[compareList[[i, 2]]]]], {i, Length[compareList]}]], 50,
 LabelingFunction → Above](*the histogram of the distances between
 all block pairs. used to determine the threshold*)
```

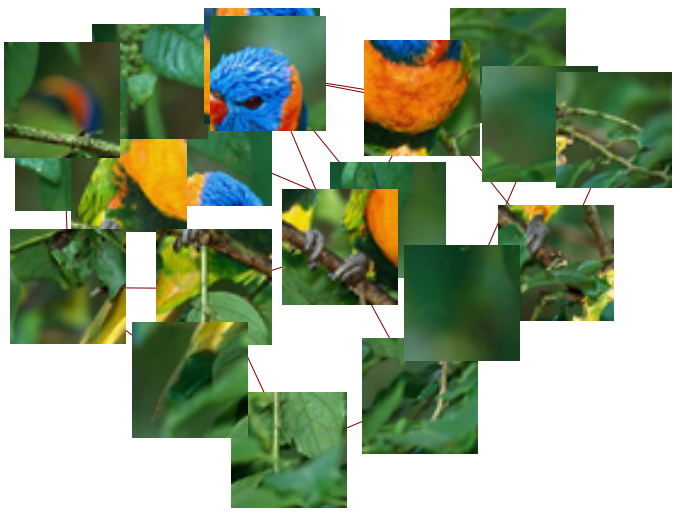

200
150
100
50
0
10
20
30
40
6 5 6 5 6
26
81
125
155
212
205
196
208
187
138
124
111
133
116
124
111
91
80
56
59
44
45
53
64
78
63
48
33
19
8
1
5 5 3 2
1
1 1

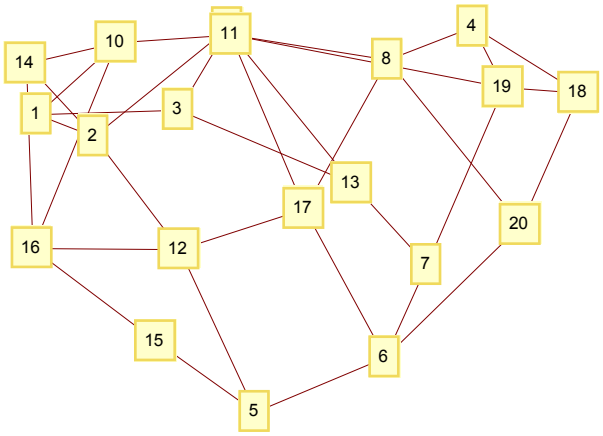
# Process raw connection pairs to find correct ones

```
threshold = 10;
rawPair = {};
Do[cl1 = compareList[[i, 1]];
  cl2 = compareList[[i, 2]];
  dist = cDist[borders[[cl1]], borders[[cl2]]];
  If[dist ≤ threshold, AppendTo[rawPair, {cl1, cl2, dist}]],
  {i, Length[compareList]}];
(*generate the raw connection pairs*)
GBF = Gather[rawPair, (#1[[1]] == #2[[1]] || #1[[1]] == #2[[2]]) &];
(*gather shortest pairs on first edge, swap 1st and 2nd if 2nd has appeared*)
minGBF = Table[Flatten[MinimalBy[GBF[[i]], Last], 1], {i, Length[GBF]}];
Dimensions[minGBF];(*show how many pairs in it*)
GBS = Gather[minGBF, (#1[[2]] == #2[[2]]) &];
(*gather shortest pairs on second edge*)
minGBS = Table[Flatten[MinimalBy[GBS[[i]], Last], 1], {i, Length[GBS]}];
Dimensions[minGBS];(*show how many pairs in it*)
GBIdx = Gather[minGBS, (Ceiling[#1[[1]] / 4] == Ceiling[#2[[1]] / 4] &&
      Ceiling[#1[[2]] / 4] == Ceiling[#2[[2]] / 4]) &];
DelExtr = Table[Flatten[MinimalBy[GBIdx[[i]], Last], 1], {i, Length[GBIdx]}];
(*delete the longer edge between two duplicated vertices pairs*)
Dimensions[DelExtr];(*show how many pairs in it*)

edgeUndirected = Table[
   Ceiling[DelExtr[[i, 1]] / 4] ⟷ Ceiling[DelExtr[[i, 2]] / 4], {i, Length[DelExtr]}];
{gridH, gridW} = {imH / pxl, imW / pxl};
nEdge = (gridH - 1) * gridW + (gridW - 1) * gridH;
(*correct edge number for a gridH by gridW puzzle*)
neighbor = Table[{}, {i, Length[block]}];
(*establish neighbor list of each vertex according to edge list*)
Do[i1 = Ceiling[DelExtr[[i, 1]] / 4];
  i2 = Ceiling[DelExtr[[i, 2]] / 4];
  AppendTo[neighbor[[i1]], i2];
  AppendTo[neighbor[[i2]], i1], {i, Length[DelExtr]}];

checkList = {};(*find possibly wrong edges*)
Table[v1 = edgeUndirected[[e, 1]]; v2 = edgeUndirected[[e, 2]];
  nbr1 = DeleteCases[neighbor[[v1]], v2];
  nbr2 = DeleteCases[neighbor[[v2]], v1];
  {v1, v2, If[
    Count[Flatten[Table[GraphDistance[Graph[DeleteCases[edgeUndirected, v1 ⟷ v2]],
          nbr1[[i]], nbr2[[j]]], {i, Length[nbr1]}, {j, Length[nbr2]}]], x_ /; x > 3] ≥
     0, AppendTo[checkList, DelExtr[[e]]]]}, {e, Length[edgeUndirected]}];
wrongEdge = {};(*MaximalBy[checkList,Last,Length[DelExtr]-nEdge];*)
finalPair = Complement[DelExtr, wrongEdge];
edgeDircted = Table[Ceiling[finalPair[[i, 1]] / 4] → Ceiling[finalPair[[i, 2]] / 4],
   {i, Length[finalPair]}];
```

# Visualization and testing

```
randSeed = RandomInteger[100];(*generate random seed for graph visualization*)
GraphPlot[edgeDircted,
 VertexRenderingFunction → (Inset[Magnify[block[[#2]], 60], #] &),
 Method → {"SpringEmbedding", "RandomSeed" → randSeed},
 VertexLabeling → True] (*plotgraph*)
GraphPlot[edgeDircted, Method → {"SpringEmbedding", "RandomSeed" → randSeed},
 VertexLabeling → True] (*plotgraph*)

map = Transpose[{Range[imH / pxl * imW / pxl], shuffleList}];
mark = Transpose[SortBy[map, Last]][[1]];
(*reverse mapping relation, marks will be shown on shuffled blocks*)
marked = Table[ImageCompose[GradientFilter[imgBlock[[i, j]], 2], Graphics[
      Style[Text[mark[[(i - 1) * partitionW + j]]], FontSize → 20, Bold, White]]],
   {i, partitionH}, {j, partitionW}];
MatrixForm@marked(*plot marked graph*)
```





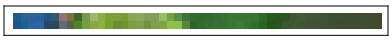

```
printPair[pair_] := (Print@Length[pair];
  (*display all information of connection pairs for analysis*)
  Column@Table[
    i1 = pair[[i, 1]];
    i2 = pair[[i, 2]];
    checkByBorderIdx[i1, i2], {i, Length[pair]}])
```

```
printPair[DelExtr]
```

35

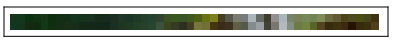
{10, 2, 11, 1, 6.83878}

{1, 2, 3, 4, 1.90791}

{1, 3, 2, 1, 2.98132}

{1, 4, 10, 2, 2.78525}

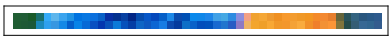
{7, 3, 19, 1, 0.592066}

{2, 2, 9, 4, 4.25535}

{2, 3, 12, 1, 5.77022}

{2, 4, 14, 2, 1.12056}

{7, 4, 13, 2, 0.670666}

{3, 2, 13, 4, 0.367906}

{3, 3, 9, 1, 6.47362}

{11, 2, 19, 4, 0.631221}

{4, 1, 19, 3, 2.66264}

{18, 1, 19, 2, 4.03618}

{4, 3, 18, 1, 0.651101}

{4, 4, 8, 2, 1.81075}

{5, 1, 12, 3, 3.38341}

{5, 2, 6, 4, 2.8777}

{5, 4, 15, 2, 1.91973}

{6, 1, 17, 3, 3.92361}

{6, 2, 20, 4, 3.62055}

{6, 3, 7, 2, 6.01763}

{8, 1, 11, 3, 4.7598}

{8, 3, 20, 1, 5.80155}

{8, 4, 17, 2, 2.86897}

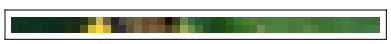
{9, 2, 11, 4, 7.14867}

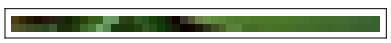
{9, 3, 17, 1, 2.97613}

{10, 1, 16, 4, 7.84101}

{10, 3, 14, 1, 1.33712}

{11, 1, 13, 3, 0.587428}

{12, 2, 17, 4, 4.76863}

{12, 4, 16, 2, 5.5448}

{14, 3, 16, 1, 6.74371}

{15, 1, 16, 3, 3.58651}

{18, 4, 20, 2, 5.5735}


# Some scratches for testing

```
Gather[DelExtr, (Ceiling[#1[[2]] / 4] == Ceiling[#2[[1]] / 4]) &]
```

```
{{{38, 41, 6.83878}, {42, 76, 0.631221}, {41, 51, 0.587428}},
 {{2, 12, 1.90791}, {10, 52, 0.367906}, {11, 33, 6.47362}},
 {{3, 5, 2.98132}, {6, 36, 4.25535}, {7, 45, 5.77022}, {8, 54, 1.12056}},
 {{4, 38, 2.78525}, {37, 64, 7.84101}, {39, 53, 1.33712}},
 {{27, 73, 0.592066}}, {{28, 50, 0.670666}}, {{13, 75, 2.66264}},
 {{69, 74, 4.03618}}, {{15, 69, 0.651101}, {72, 78, 5.5735}},
 {{16, 30, 1.81075}, {29, 43, 4.7598}, {31, 77, 5.80155}, {32, 66, 2.86897}},
 {{17, 47, 3.38341}, {46, 68, 4.76863}, {48, 62, 5.5448}},
 {{18, 24, 2.8777}, {21, 67, 3.92361}, {22, 80, 3.62055}, {23, 26, 6.01763}},
 {{20, 58, 1.91973}, {57, 63, 3.58651}}, {{34, 44, 7.14867}},
 {{35, 65, 2.97613}}, {{55, 61, 6.74371}}}
```