Super Resolution for Automated Target Recognition

Maksim Levental

November 29, 2019

Abstract

Super resolution is the process of producing high-resolution images from low-resolution images while preserving ground truth about the subject matter of the images and potentially inferring more such truth. Algorithms that successfully carry out such a process are broadly useful in all circumstances where high-resolution imagery is either difficult or impossible to obtain. In particular we look towards super resolving images collected using longwave infrared cameras since high resolution sensors for such cameras do not currently exist. We present an exposition of motivations and concepts of super resolution in general, and current techniques, with a qualitative comparison of such techniques. Finally we suggest directions for future research.

1 Appendix

1.1 Automatic Differentiation

Let

$$F\colon \boldsymbol{x} \in \mathbb{R}^n \mapsto y \in \mathbb{R}$$

and decompose F as

$$F = D \circ C \circ B \circ A$$

where

$$\begin{aligned} A\colon \boldsymbol{x} &\in \mathbb{R}^n \mapsto \boldsymbol{a} \in \mathbb{R}^m \\ B\colon \boldsymbol{a} &\in \mathbb{R}^m \mapsto \boldsymbol{b} \in \mathbb{R}^k \\ C\colon \boldsymbol{b} &\in \mathbb{R}^k \mapsto \boldsymbol{c} \in \mathbb{R}^j \\ D\colon \boldsymbol{c} &\in \mathbb{R}^j \mapsto y \in \mathbb{R} \end{aligned}$$

Let $y_0 = F(\boldsymbol{x}_0)$ where $\boldsymbol{x}_0 \coloneqq (x_{01}, \dots, x_{0n})$. Then the gradient of F evaluated at $\boldsymbol{x}_0$ is

$$F'(\boldsymbol{x}_0) = \left.\frac{\partial y}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_0} = \left(\left.\frac{\partial y}{\partial x_1}\right|_{x_1=x_{01}}, \dots\dots, \left.\frac{\partial y}{\partial x_n}\right|_{x_n=x_{0n}}\right)$$

Note we are implicitly defining the parameter of $F' = \frac{\partial y}{\partial \boldsymbol{x}}$ to be $\boldsymbol{x}$. This is fine because parameters can be renamed at will[1] but it's important to keep in mind that F' is wholly different from F and could have any other parameter. By the chain rule

$$F'(\boldsymbol{x}_0) = D'(C'(B'(A'(\boldsymbol{x}_0))))$$

or if we define intermediate values

$$\begin{aligned} \boldsymbol{a}_0 &= A(\boldsymbol{x}_0) \\ \boldsymbol{b}_0 &= B(\boldsymbol{a}_0) \\ \boldsymbol{c}_0 &= C(\boldsymbol{b}_0) \\ y_0 &= D(\boldsymbol{c}_0) \end{aligned}$$

[1] Function parameters are *bound variables*. The function definition *binds* the variable.

then

$$F'(\boldsymbol{x}_0) = \left.\frac{\partial y}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\boldsymbol{c}_0} \times \left.\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{b}}\right|_{\boldsymbol{b}=\boldsymbol{b}_0} \times \left.\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{a}}\right|_{\boldsymbol{a}=\boldsymbol{a}_0} \times \left.\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_0} \tag{1}$$

where now a term like $\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{a}}|_{\boldsymbol{a}=\boldsymbol{a}_0}$ is the *Jacobian* of B evaluated at $\boldsymbol{a}_0$:

$$\begin{aligned}\left.\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{a}}\right|_{\boldsymbol{a}=\boldsymbol{a}_0} &= \left.\begin{bmatrix} \frac{\partial b_1}{\partial a_1} & \cdots & \frac{\partial b_1}{\partial a_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial b_k}{\partial a_1} & \cdots & \frac{\partial b_k}{\partial a_m} \end{bmatrix}\right|_{\boldsymbol{a}=\boldsymbol{a}_0} \\ &\coloneqq \begin{bmatrix} \left.\frac{\partial b_1}{\partial a_1}\right|_{a_1=a_{01}} & \cdots & \left.\frac{\partial b_1}{\partial a_m}\right|_{a_m=a_{0m}} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial b_k}{\partial a_1}\right|_{a_1=a_{01}} & \cdots & \left.\frac{\partial b_k}{\partial a_m}\right|_{a_m=a_{0m}} \end{bmatrix}\end{aligned}$$

The right hand side (RHS) of eqn. (1) is associative; if we want to compute F' in general we can accumulate products starting from the right or starting from the left:

$$F' = \frac{\partial y}{\partial \boldsymbol{c}}\left(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{b}}\left(\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{a}}\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{x}}\right)\right) \tag{2}$$

$$F' = \left(\left(\frac{\partial y}{\partial \boldsymbol{c}}\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{b}}\right)\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{a}}\right)\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{x}} \tag{3}$$

Computing F' as in eqn. (2) is called *forward accumulation* (because it parallels the evaluation order of F) and, conversely, deriving F' as in eqn. (3) is called *reverse accumulation*. Notice that since $F\colon \mathbb{R}^n \to \mathbb{R}$ it's the case that while $\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{a}} \cdot \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{x}}$ is a jacobian–jacobian product

$$\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{a}} \cdot \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial b_1}{\partial a_1} & \cdots & \frac{\partial b_1}{\partial a_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial b_k}{\partial a_1} & \cdots & \frac{\partial b_k}{\partial a_m} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial a_1}{\partial x_1} & \cdots & \frac{\partial a_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_m}{\partial x_1} & \cdots & \frac{\partial a_m}{\partial x_n} \end{bmatrix} \tag{4}$$

while $\frac{\partial y}{\partial \boldsymbol{c}} \cdot \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{b}}$ is a vector–Jacobian product (called a VJP)

$$\frac{\partial y}{\partial \boldsymbol{c}} \cdot \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{b}} = \left(\frac{\partial y}{\partial c_1}, \ldots, \frac{\partial y}{\partial c_j}\right) \cdot \begin{bmatrix} \frac{\partial c_1}{\partial b_1} & \cdots & \frac{\partial c_1}{\partial b_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial c_j}{\partial b_1} & \cdots & \frac{\partial c_j}{\partial b_k} \end{bmatrix} \tag{5}$$

All intermediate products will also be VJPs (for reverse accumulation); in general reverse accumulation is more efficient when the dimension of the domain is greater than the dimension of the range.

1.2 Splines

A *spline* is a piecewise defined polynomial function. For example a simple *order* $d+1$ spline s could be defined

$$s(x) \coloneqq \sum_{i=0}^{d} \mathbb{1}_{[k_j, k_{j+1})} P_{ij} \cdot (x - k_j)^i$$

where the $n+1$ increasing *knots* $k_0 < \cdots < k_n$ bracket n intervals over which the component polynomials are defined, and the $(d+1)$ polynomial coefficients P_{ij} define the spline (see figure 1) for $x \in [k_j, k_{j+1})$ ($\mathbb{1}$ enforces this[2]). Splines need not be differentiable (nor even continuous) at the knots but can be specified with such continuity constraints (at the knots). The number of degrees of freedom of a spline over $n+1$ knots and of order $d+1$ (i.e., the dimension of the vector space[3] of such splines) is the number of polynomial coefficients minus the number of continuity conditions. For example if the spline is constrained to be maximally continuous (i.e., continuous and differentiable $d-1$ times[4]) then the spline has $d+1$ polynomial coefficients for every one of the n knot intervals and d continuity constraints at every one of the $n-1$ interior knots (continuity constraints cannot be specified at the boundary knots), which implies

$$n(d+1) - (n-1)d = n + d$$

degrees of freedom. The spline construction naturally extends to dimensions greater than 1.

A Basic-spline (B-spline) is a spline defined as a linear combination of a particular set of basis functions: the B-spline basis element $B_{i,d+1}$ of order $d+1$ (degree d) over knots $k_i < \cdots < k_{i+d+1}$ is defined recursively according to the Cox-de Boor recursion formula **de1971subroutine**:

$$B_{j,1}(x) \coloneqq \mathbb{1}_{[k_j, k_{j+1})} \tag{6}$$

$$B_{j,d+1}(x) \coloneqq \frac{x - k_j}{k_{j+d} - k_j} B_{j,d}(x) \tag{7}$$

$$+ \frac{k_{j+d+1} - x}{k_{j+d+1} - k_{i+1}} B_{j+1,d}(x) \tag{8}$$

Note that the coefficients

$$\frac{x - k_j}{k_{j+d} - k_j}$$

[2]The indicator (or characteristic) $\mathbb{1}_{[a,b)}$ function is 1 on the interval $[a, b)$ and 0 otherwise.

[3]A set whose members (*vectors*) can be decomposed as linear combinations of elementary elements (a *basis*).

[4]Note that even though a d-degree polynomial has d derivatives we only require agreement at the first $d-1$ derivatives (and continuity itself). This is owing to the fact that if we required all d derivatives to agree the number of degrees of freedom would be $(n(d+1) - (n-1)(d+1) = d+1$ and therefore the spline would no longer be a piecewise polynomial but simply a polynomial (of degree d).

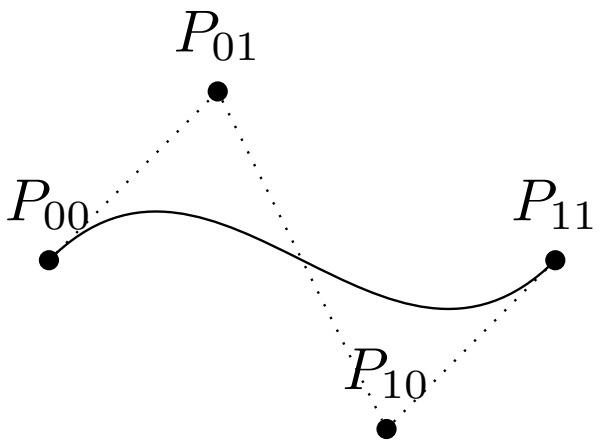


Figure 1: Cubic spline.

and

$$\frac{k_{j+d+1} - x}{k_{j+d+1} - k_{i+1}}$$

interpolate smoothly between the lower order B-splines $B_{j,d}, B_{j+1,d}$ as x goes from k_j to k_{j+d+1}.

1.3 Delaunay Triangulation

Delaunay triangulation for a set points in the plane is a such that no point in is inside the circumcircle of any triangle in triangulation (see figure 2). A local

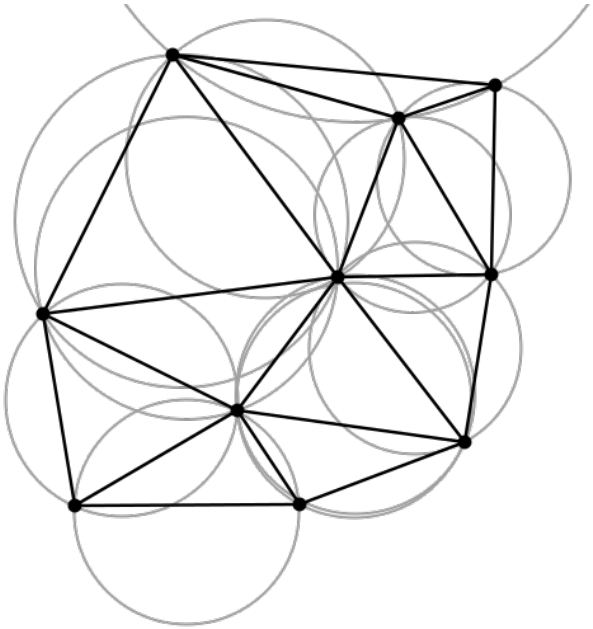
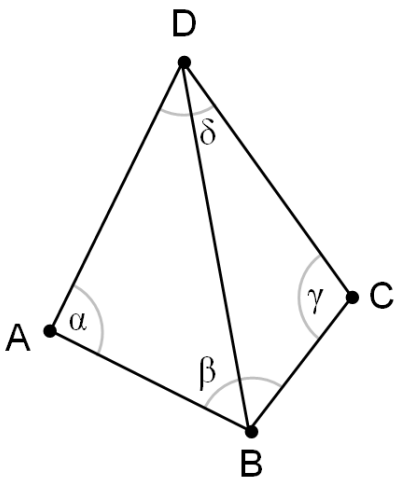
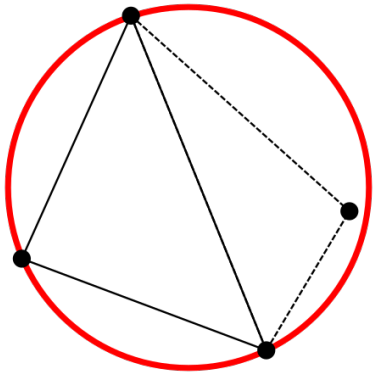


Figure 2: Delaunay Triangulation.

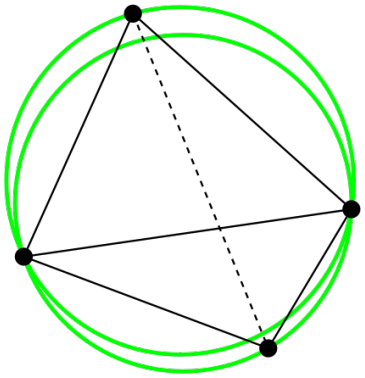
characterization two triangles ABD and BCD with the common edge BD (see figure 3a), if the sum of the angles α and γ is less than or equal to 180°, the triangles meet the Delaunay condition. This is an important



(a) This triangulation does not meet the Delaunay condition (the sum of α and γ is bigger than 180°).



(b) This pair of triangles does not meet the Delaunay condition (the circumcircle contains more than three points).



(c) Flipping the common edge produces a valid Delaunay triangulation for the four points.

Figure 3: Visual Delaunay definition and Flipping.

property because it allows the use of a flipping technique. If two triangles do not meet the Delaunay condition, switching the common edge BD for the common edge AC produces two triangles that do meet the Delaunay condition (see figures 3b, 3c). This leads to a straightforward algorithm: construct any triangulation of the points, and then flip edges until no triangle is non-Delaunay.

Index