Super Resolution for Automated Target Recognition

Maksim Levental

***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

1.0.1 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

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

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 $\left.\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{a}}\right|_{\boldsymbol{a}=\boldsymbol{a}_0}$ is the *Jacobian* of B evaluated at $\boldsymbol{a}_0$:

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

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

$$\coloneqq \begin{bmatrix} \frac{\partial b_1}{\partial a_1}\Big|_{a_1=a_{01}} & \cdots & \frac{\partial b_1}{\partial a_m}\Big|_{a_m=a_{0m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial b_k}{\partial a_1}\Big|_{a_1=a_{01}} & \cdots & \frac{\partial b_k}{\partial a_m}\Big|_{a_m=a_{0m}} \end{bmatrix}$$

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.

1.0.2 Pushforwards and Pullbacks

Define a diffeomorphism ϕ such that

$$\phi\colon (r, \theta) \mapsto (r\cos\theta, r\sin\theta) \tag{6}$$

The Jacobian of ϕ

$$\mathrm{J} = \begin{bmatrix} \frac{\partial \phi_1}{\partial r} & \frac{\partial \phi_1}{\partial \theta} \\ \frac{\partial \phi_2}{\partial r} & \frac{\partial \phi_2}{\partial \theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{bmatrix} \tag{7}$$

Note that $\det \mathrm{J} = r$ and so ϕ is a diffeomorphism iff $r \neq 0$. Given a vector field

$$v = a(r,\theta)\partial_r + b(r,\theta)\partial_\theta \coloneqq a(r,\theta)\frac{\partial}{\partial r} + b(r,\theta)\frac{\partial}{\partial \theta} \tag{8}$$

we can compute the *pushforward* ϕ_* wrt the ∂_x, ∂_y basis

$$\phi_*(v) = \begin{bmatrix} \cos(\theta) & -r\sin(\theta) \\ \sin(\theta) & r\cos(\theta) \end{bmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a\cos(\theta) - br\sin(\theta) \\ a\sin(\theta) + br\cos(\theta) \end{pmatrix} \tag{9}$$

Hence, explicitly

$$\phi_*(v) = (a\cos(\theta) - br\sin(\theta))\partial_x + (a\sin(\theta) + br\cos(\theta))\partial_y \tag{10}$$

Since J is invertible we can investigate which vector fields map to ∂_x

$$\phi_* v = \partial_x \iff v = \phi_*^{-1}\partial_x \tag{11}$$

Let $v = a\partial_r + b\partial_\theta$. Then

$$v = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\frac{\sin(\theta)}{r} & \frac{\cos(\theta)}{r} \end{bmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos(\theta) \\ -\frac{\sin(\theta)}{r} \end{pmatrix} \tag{12}$$

However we need to write r, θ in terms of x, y

$$\phi_*^{-1}\partial_x = \frac{x}{\sqrt{x^2+y^2}}\partial_r + \frac{y}{x^2+y^2}\partial_\theta \tag{13}$$

ϕ_*^{-1} is called the *pullback* ϕ^* of the vector field ∂_x along ϕ.