UNIVERSITY OF CALIFORNIA
SANTA CRUZ

GAUSSIAN PROCESS MODELING FOR UPSAMPLING ALGORITHMS WITH APPLICATIONS IN COMPUTER VISION AND COMPUTATIONAL FLUID DYNAMICS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

APPLIED MATHEMATICS AND STATISTICS

by

Steven I Reeves

March 2020

The Dissertation of Steven I Reeves
is approved:

Table of Contents

List of Figures

List of Tables

Abstract

Gaussian Process Modeling for Upsampling Algorithms with Applications in Computer Vision and Computational Fluid Dynamics

by

Steven I Reeves

Across a variety of fields, interpolation algorithms have been used to upsample low resolution or coarse data fields. In this work, novel Gaussian Process based methods are employed to solve a variety of upsampling problems. Specifically three applications are explored: coarse data prolongation in Adaptive Mesh Refinement (AMR) in the field of Computational Fluid Dynamics, accurate document image upsampling to enhance Optical Character Recognition (OCR) accuracy, and fast and accurate Single Image Super Resolution (SISR). For AMR, a new, efficient, and "3rd order accurate" algorithm called GP-AMR is presented. Next, a novel, non-zero mean, windowed GP model is generated to upsample low resolution document images to generate a higher OCR accuracy, when compared to the industry standard. Finally, a hybrid GP convolutional neural network algorithm is used to generate a computationally efficient and high quality SISR model.

This body of work is dedicated to my wife, Erika, who has been a constant pillar of support through the trials of graduate education and research.

Acknowledgments

The Adaptive Mesh Refinement chapter of this dissertation has come to fruition with help from the Center for Computational Science and Engineering at Lawrence Berkeley National Laboratory, the home of AMReX. I thank Dr. Ann S. Almgren and Dr. Weiqun Zhang for their insight and advice when completing this work. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. Additionally, I acknowledge use of the Lux supercomputer at the University of California, Santa Cruz, funded by NSF MRI grant AST 1828315.

The upsampling method for Optical Character Recognition was tested on issue 2402 of Le Nouvel Observateur. These pages are available for reuse for research purposes in evaluation and illustration. This data is copyrighted, ©2012 EPITA Research and Development Laboratory (LRDE) with permission from Le Nouvel Observateur. I thank Thierry Géraud and Le Nouvel Observateur for making this data publicly available for research.

I thank my advisor, Dr. Dongwook Lee, for giving me the mathematical tools to tackle these problems that have, at first glance, little in common. Additionally, I give thanks to Dr. Nicholas Brummell and Dr. Roberto Manduchi for forming my committee and giving validity to my work.

Furthermore, I am grateful for the knowledge and expertise of Dr. Anurag Singh, who gave his much needed advice in the field of computer vision and optical character recognition. Lastly, I thank Tyler Whitlock for lending me an extra GPU when access to large compute systems were unavailable.

Chapter 1

Introduction

Generating new data from sampled data is a prevalent operation in the sciences and in various industrial applications. In this dissertation, new methods for upsampling are explored for applications in Adaptive Mesh Refinement (AMR) for Computational Fluid Dynamics (CFD), Optical Character Recognition (OCR) and Single-Image Super Resolution (SISR) in Computer Vision (CV).

In any data based application, finite samples are measured either by simulation, experimentation or observation. These samples represent values of an unknown function for a limited number of independent variable instances. Interpolation, or upsampling, is the process of estimating the values an unknown function would yield given intermediate independent variable data. In many cases, the unknown function can be non-trivial or not representable in closed form, e.g. the solution to nonlinear partial differential equations. Interpolation gives an estimate of this complicated function using bases comprised of somewhat simpler functions, while producing a result that is fairly close to the real solution. Polynomial based interpolation methods have been widely explored in various domains of science and are often the baselines when researching into interpolation or upsampling. In this dissertation, however, non-linear non-polynomial approaches utilizing Gaussian Process Modeling are used for the aforementioned applications.

This dissertation is organized by application. The rest of chapter 1 introduces Gaussian Processes as a whole comprising section 1.1. In Chapter 2, two different Gaussian Process based interpolation methods are presented for the express purposes of generating new mesh data from lower resolution computational meshes. Chapter 2 also introduces the computational framework in which these algorithms are implemented. This chapter culminates in a comparison of commonly used tests for compressible hydrodynamics simulation codes.

For chapters 3, 4, GP based algorithms are produced for upsampling image data. In chapter 3, a standalone GP algorithm is used to upsample low resolution grayscale documents for the purposes of enhancing optical character recognition. This chapter discusses the differences in covariance kernel used to build the Gaussian Process Model and introduces the use of a maximum likelihood estimate for the mean. Finally, a comparison of the baseline bicubic interpolation and the GP model is performed on the 2402 issue of Le Nouvel Observateur.

In chapter 4 a hybrid Gaussian Process Deep Learning model is used to generate realistic looking images from low resolution examples. This chapter features a simplified version of the GP model generated in chapter 3, the sample pixel window is reduced from a 5×5 pixel patch to a 3×3 pixel patch, and the maximum likelihood estimate for the mean is not used. This simplified model is compared to the baseline upsampling method bicubic. However, the state-of-the-art for super-resolution is more than these baseline upsampling methods. So to meet this need, a convolutional neural network is constructed to enhance the upsampled image delivered by the GP method. This creates a fast and high quality super resolution pipeline that challenges more complex and computationally intensive procedures.

Finally, chapter 5 summarizes the works presented in the previous chapters and offers a final discussion.

1.1 Gaussian Process Modeling

The methods proposed in this dissertation are all fundamentally based on Gaussian Process modeling and regression. The topic of GP has a wide breadth and is a fundamental tool in statistics. This section offers a brief overview on Gaussian Process modeling and constructing a Gaussian Process. Gaussian Processes are a family of stochastic processes such that any finite collection of random variables sampled from these processes are joint normally distributed.

In a more general sense, Gaussian Processes sample functions from an infinite dimensional function space. However, this function is not explicitly generated but rather predicts function values are prescribed points. This is the basis of the interpolating routines described in detail in sections 2.1,3.2, 4.2. The functions values that the methods describe are be drawn from a data-informed distribution spaces trained on the sampled (low resolution) data.

To construct a Gaussian Process model, one needs to specify a *prior probability distribution* for the function space. Samples (function values evaluated at known locations) are then used to update this prior probability distribution. Through the use of Bayes' Theorem the *posterior probability distribution* is generated given the prior and the samples [155, 24].

In general, the construction of the posterior probability distribution over the function space is the heart of Gaussian Process Modeling and to generate an interpolating model, functions are drawn from a data-adjusted function space. Specifically, the posterior may be used to probabilistically predict the value of a function at points where the function has not been previously sampled, i.e. the upsampling predictions.

Gaussian Processes can be fully defined by two functions: a mean function $\bar{f}(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and a covariance function that generates a symmetric, positive-definite covariance kernel $K(\mathbf{x}, \mathbf{y}) : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$. Functions, f, that are drawn from a GP with mean function $\bar{f}(\mathbf{x})$ and covariance $K(\mathbf{x}, \mathbf{y})$ are denoted as $f \sim \mathcal{GP}(\bar{f}, K)$. This is analogous to finite-dimensional distributions.

The covariance function is defined as

$$K(\mathbf{x}, \mathbf{y}) = \mathbb{E}\left[\left(f(\mathbf{x}) - \bar{f}(\mathbf{x})\right)\left(f(\mathbf{y}) - \bar{f}(\mathbf{y})\right)\right] \tag{1.1}$$

where expectation, $\mathbb{E}$, is with respect to the Gaussian Process.

One controls the GP by specifying both $\bar{f}(\mathbf{x})$ and $K(\mathbf{x}, \mathbf{y})$, typically as some hyper-parameterized functions. These hyper-parameters allow us to give the "character" of functions generated by the posterior (i.e. length scales, differentiability).

Suppose a Gaussian Process is given along with N locations or independent variables $\mathbf{x}_n \in \mathbb{R}^d, d = 1, 2, 3$ and $n = 1, \ldots, N$ at which samples $f(\mathbf{x}_n)$ are collected. Then the likelihood $\mathcal{L}$ can be calculated– the probability of the data given the GP model. Let $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^T$ be the sampled data, then the Gaussian Process likelihood function is

$$\mathcal{L} \equiv P\big(\mathbf{f} | \mathcal{GP}(\bar{f}, K)\big) = (2\pi)^{-N/2} \det |\mathbf{K}|^{-1/2} \exp\left[-\frac{1}{2}\left(\mathbf{f} - \bar{\mathbf{f}}\right) \mathbf{K} \left(\mathbf{f} - \bar{\mathbf{f}}\right)\right]. \tag{1.2}$$

Here $\mathbf{K}$ is the covariance kernel matrix generated by evaluating the covariance function using the independent variable data in the sample, mathematically: $K_{n,m} = K(\mathbf{x}_n, \mathbf{x}_m)$ where $n, m = 1, \ldots, N$. Furthermore, $\bar{\mathbf{f}} = [\bar{f}(\mathbf{x}_1), \cdots \bar{f}(\mathbf{x}_N)]$. Using these samples, a probabilistic statement is made about the value of the function $f \sim \mathcal{GP}(\bar{f}, K)$ at a new unsampled point $\mathbf{x}_*$. That is, a model for the value of $f(\mathbf{x}_*)$ is generated using this Gaussian Process. This portion is especially important for the applications in this dissertation, as new data needs to be constructed at finer levels or higher resolution.

Applying Bayes' Theorem with the conditioning property, directly onto the joint Gaussian prior given the samples $\mathbf{f}$ give the posterior distribution of the predicted value, f_* given $\mathbf{f}$,

$$P(f_*|\mathbf{f}) = (2\pi U^2)^{-1/2} \exp\left[-\frac{(f_* - \bar{f}_*)^2}{2U^2}\right]. \tag{1.3}$$

Essential for these applications, the posterior PDF gives a new *posterior mean function*

$$\tilde{f}_* \equiv \bar{f}(\mathbf{x}_*) + \mathbf{k}_*^T \mathbf{K}^{-1} \cdot (\mathbf{f} - \bar{\mathbf{f}}), \tag{1.4}$$

and *posterior covariance*

$$U^2 \equiv k_{**} - \mathbf{k}_*^T \mathbf{K}^{-1} \cdot \mathbf{k}_*. \tag{1.5}$$

This shows that Gaussian Process predictions are not only estimates for the upsampling value, but also has uncertainty information [155]. The uncertainty information for is categorized as the marginal distribution at the point of prediction and manifests itself as a Gaussian distribution or Multivariate Gaussian for multi-output predictions [35]. The uncertainty characterizaiton is what the posterior covariance is used for.

However, in the applications presented in this dissertation, the uncertainty portion of the Gaussian Process will not be used, as an informed interpolation algorithm will be generated for set upsampling factors. That said, the uncertainty measure could be feasibly used to detect regions of high uncertainty for an adaptive upsampling algorithm, where regular intervals of data are not required. None-the-less, for these applications, the posterior mean will be used alone as an upsampling model.

Chapter 2

Applications of GP for Adaptive Mesh Refinement

Since the dawn of the computer era for science and engineering, the primary role of computational fluid dynamics (CFD) is to advance our theoretical understandings. Through experimentation, a wide range of parameter spaces is designed and modeled in computer simulations. As such, computer-aided simulations target complex physical conditions in various degrees of disparity adequate to users' specific theoretical models. As increasingly more complex systems to be considered for better computer modeling, modern simulation codes face increasingly versatile challenges to meet expected metrics in a possibly vast parameter space. These complex simulations need to be able to be interpreted as *physically valid* models, at least in an approximated sense.

The fields of geophysics, astrophysics, and laboratory plasma astrophysics are good exampled where computer simulations are essential (e.g., [73, 101, 211, 135]). CFD has been (and will continuously be) an indispensable tool to improve our capabilities to investigate far-reaching research expeditions in these fields of study. In certain physical scenarios, the experiment could develop into a flow condition in which the required physics become extremely challenging to simulate due to the great imbalance in length and temporal scales. To alleviate such conditions in computer simulations, practitioners have explored approaches by which a computer simulation can focus on localized flow regions when the dynamics exhibit confined features that evolve on a much shorter length scale relative to the flow dynamics on the rest of the computational domain.

Adaptive mesh refinement (AMR) is one such approach that allows a local and dynamic change in the grid resolutions of a simulation in space and time. Since the 1980s, AMR has been an exceptional tool and has become a powerful strategy in utilizing CFD simulations for computational science across many disciplines such as astrophysics, geophysics, atmospheric sciences, oceanography, biophysics, engineering, and many others [150].

There have been many advancements in AMR since the seminal paper by Berger and Oliger [22]. In their paper, the primary concern was to focus on a strategy for generating subgrids and managing the grid hierarchy for scalar hyperbolic PDEs in one and two spatial dimensions (1D and 2D). In the subsequent work by Berger and Colella [21], further improvements were made possible for numerical solutions of the 2D Euler equations to provide a robust shock-capturing AMR algorithm that satisfies the underlying conservation property in on large-scale computer architectures. The novel innovations in their work have now become the AMR standards, namely including refluxing (or flux correction) between fine-coarse interface boundaries, conservative (linear) prolongation and restriction on AMR hierarchies, and timestep subcycling. Bell *et al.* extended the precedent 2D AMR algorithms of [22, 21] to a 3D AMR algorithm and applied it to solve 3D hyperbolic systems of conservation laws [20]. They demonstrated that the AMR algorithm reduced the computational cost by more than a factor of 20 than on the equivalent uniform grid simulations in simulating a 3D dense cloud problem interacting a Mach 1.25 flow on Cray-2. This is, by far, the main benefit of using AMR, particularly in large 3D simulations, in that one could gain such a computational speed-up by exercising the computational efficiency as a consequence of allowing higher-resolution calculations only locally where needed.

Jameson [98] examined computational gain with AMR further. Simulations using the traditional second-order AMR schemes (i.e., second-order PDE solutions solved on AMR grids) could become computationally more expensive than the compared uniform grid (non-AMR) simulations using high-order (4th or higher) PDE solvers, particularly when a large fraction of the computational domain contains fine scale structures such as vortices, eddies, rotating flows, turbulence, etc. In this case, one should rely on high-order PDE solvers on uniform grids to get the best computational results, by which small scale flow features are better resolved on a given "static" grid than the second-order AMR calculation. According to Jameson's estimation, the amount of the flow of interests such as shocks, vortices, and other small scale flows should not exceed more than 1/3 of the computational domain in order that low order AMR schemes become computationally competitive. Also see the study on the effectiveness of AMR in atmospheric simulations [68]. Of importantly related is the consistency between calculations on AMR and uniform grids. Mathematically speaking, AMR calculations should converge to the corresponding uniform grid solutions in the limit of grid convergence; otherwise exercising AMR for the purpose of gaining computational efficiency becomes no avail. In a numerical comparison study, Schmidt *et al.* [172] considered conditions upon which statistical agreement can be achieved between AMR and uniform grid calculations at the same effective grid resolutions.

There exist a variety of different approaches in modern AMR implementations. Two main types of AMR can be categorized into structured and unstructured. Unstructured AMR, and meshes in general, are very useful for problems with irregular geometry (e.g., many structural engineering problems), but is often computationally complex and difficult to handle when regridding. On the other hand, structured AMR (SAMR, or block-structured AMR) offers practical benefits (over unstructured) such as ease of discretization, a global index space, accuracy gain through cancellation terms, and ease of parallelization.

In block-structured AMR, the solution to a PDE is constructed on a hierarchy of levels with different resolution. Each level is composed of a union of logically rectangular grids or patches. These patches can change dynamically throughout a simulation. In general, patches need not be fixed size, and may not have one unique parent grid. Figure 2.1 illustrates the use of AMR in a block-structured environment.

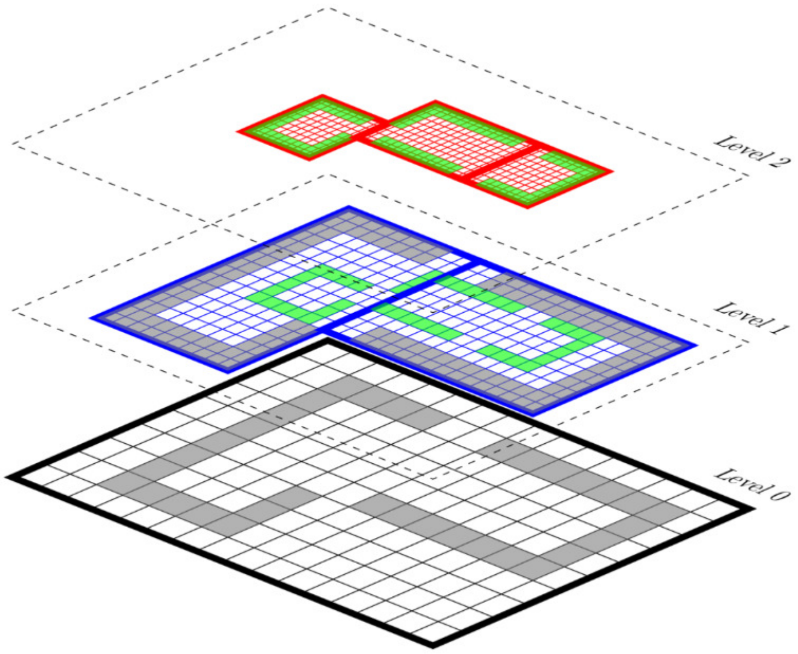


Figure 2.1: Multiple levels for block-structured AMR grid hierarchy

The approach presented by Berger and Oliger [22] and Berger and Colella [21] has set the foundation on the patch-based SAMR. An alternative to the patch-based formulation is the octree-based approach which has evolved into the fully-threaded tree (FTT) formalism (or cell-based) of Khokhlov [107] and the block-based octree of MacNiece *et al.* [131] & van der Holst *et al.* [212].

Such AMR methods have gained popularity over the past 30 years and have been adopted by various codes in astrophysics. Some of the well-known examples implementing the patch-based AMR include AstroBEAR [45], ENZO [28], ORION [109], PLUTO [138], CHARM [139], CASTRO [9], MAESTRO [145]; the octree-based AMR has been implemented in FLASH [70, 56], NIRVANA [227], BATS-R-US [151, 74]; the FTT AMR in RAMSES [197], ART [111]. The AMRVAC code [105] features both the patch-based and octree-based AMR schemes.

In contrast to these codes that incorporate AMR with the purpose of delivering specific applications in astrophysics, other frameworks have pursued a more general functionality. Examples include PARAMESH [131] which supplies solely the octree-based block-structured mesh capability independent of any governing equations; AMReX [224] is another a standalone grid software library that provides the patch-based SAMR support; Chombo [39, 4] and SAMRAI [92], on the other hand, supply both AMR capabilities and a more broader support for solving general systems of equations of hyperbolic, parabolic, and elliptic partial differential equations (PDEs). A more compressive survey on the block-structured AMR frameworks can be found in [55].

Recently, there have been many noticeable efforts aimed at designing high-order accurate solvers for governing systems of equations (e.g., [156, 134, 221, 29, 133, 64, 15, 186, 159, **?**]) in accordance with a trend of decreasing memory per compute core in newer high-performance computing (HPC) architecture [11, 54, 193]. Such high-order (4th or higher) PDE solvers are then combined with the AMR strategies described above.

Traditionally, a second-order linear interpolation scheme has been commonly adopted for data prolongation from coarse to finer AMR levels, and a mass-conserving averaging scheme for data restriction from finer to coarser levels. This "low-order" AMR interpolation model has been the default choice in the vast majority of the aforementioned AMR paradigms and algorithms in practice. The accuracy gap between the underlying high-order PDE solvers and the second-order AMR interpolation could potentially degrade the quality of solutions from the high-order PDE solvers when the solutions are projected to AMR grids that are progressively undergoing refinements and de-refinements. In addition, another accuracy loss inevitably happens at fine-coarse boundaries. It is therefore natural to close the accuracy gap in the direction of providing high-order models in AMR interpolations, to serves better to maintain the overall solution accuracy as integrated as a whole on AMR grid configurations. The high-order AMR prolongations of Shen *et al.* [181] and Chen *et al.* [34] are in this vein. These authors coupled high-order finite difference method (FDM) PDE solvers with fourth- or fifth-order accurate prolongations based on the well-known high-order polynomial interpolation schemes of WENO [175] and MP5 [194], respectively. These studies have shown that the AMR simulations with a higher-order coupling can produce better results in terms of increasing solution accuracy and lowering numerical diffusion, thereby, resolving fine-scale flow features.

The present work focuses on developing a new high-order polynomial-free interpolation scheme for AMR data prolongation on the block-structured AMR implementation using the AMReX library. Our high-order prolongation scheme stems from the previous studies on applying Gaussian Process Modeling [155] in designing high-order reconstruction/interpolation in finite volume method (FVM) [?] and in finite difference method (FDM) [?].

This chapter is organized as follows. In Section 2.0.1 the AMR framework, AMReX, where the method presented is implemented, is overviewed. Sections 2.0.2 and 2.0.3 illustrate the two Gaussian Process based methods for pointwise and cell averaged contexts. Following this, step-by-step execution details of the GP volume average algorithm are in Section 2.1. Also, a description on extending this work to a GPU-friendly implementation by following AMReX programming directives is outlined. Section 2.2 show the code performance of the new GP prolongation on selected multidimensional test problems, and finally, in Section 2.3 a summary of the main results of this work is presented.

2.0.1 AMReX

Developed and managed by the Center for Computational Science and Engineering at Lawrence Berkeley National Laboratory, AMReX is funded through the Exascale Computing Project (ECP) as a software framework to support the development of block-structured AMR applications focusing on current and next-generation architectures [224]. AMReX provides support for many operations involving adaptive meshes including multilevel synchronization operations, particle and particle/mesh algorithms, solution of parabolic and elliptic systems using geometric and algebraic multigrid solvers, and explicit/implicit mesh operations. As part of an ECP funded project, AMReX takes the hybrid MPI/OpenMP CPU parallelization along with GPU implementations (CUDA). AMReX is mostly comprised of source files that are written in C++ and Fortran. Fortran is solely used for mathematics drivers, while C++ is used for I/O, flow control, memory management and mathematics drivers.

The novelty of the current study is the new GP-based prolongation method implemented within the AMReX framework. The GP implementation furnishes an optional high-order prolongation method from coarse to fine AMR levels, alternative to the default second-order linear prolongation method in AMReX. In this way, the GP results in Section 2.2 naturally inherit all the generic AMReX operations such as load balancing, guardcell exchanges, refluxing, AMR data and grid managements, except for the new GP prolongation method.

AMR restriction is another important operation on the AMR data management in the opposite direction, from fine to coarse levels. The default restriction method of averaging down the fine grid data maintains conservation on AMR grid hierarchies. This approach populates data on coarse levels by averaging down the corresponding fine level data according to

$$\mathbf{U}^C = \frac{1}{R}\sum_i^R \mathbf{U}_i^f, \tag{2.1}$$

where $\mathbf{U}^C$ and $\mathbf{U}^f$ are conservative quantities on the coarse and fine grids respectively, $R = \prod_d r_d$ is the normalization factor with r_d being the refinement ratio in each direction $d = x, y, z$.

Lastly, maintaining conservation across fine-coarse interface levels is done by the operation called the refluxing. This process corrects the coarse grid fluxes by means of replacing them with the fluxes computed on the fine grids abutting the coarse grid. In practice, the conservation is managed as a posterior correction step after all fluid variables $\mathbf{U}^C$ on a coarse cell are updated. For example, let's consider a 2D scenario in Fig. 2.2 where there is a fine-coarse interface across an x-face whose area is given as $A^C = \Delta y$. For illustration purposes assume that there is no other refinement jump across the other three faces surrounding the coarse cell. The coarse cell data $\mathbf{U}^C$ is first updated using all four face fluxes

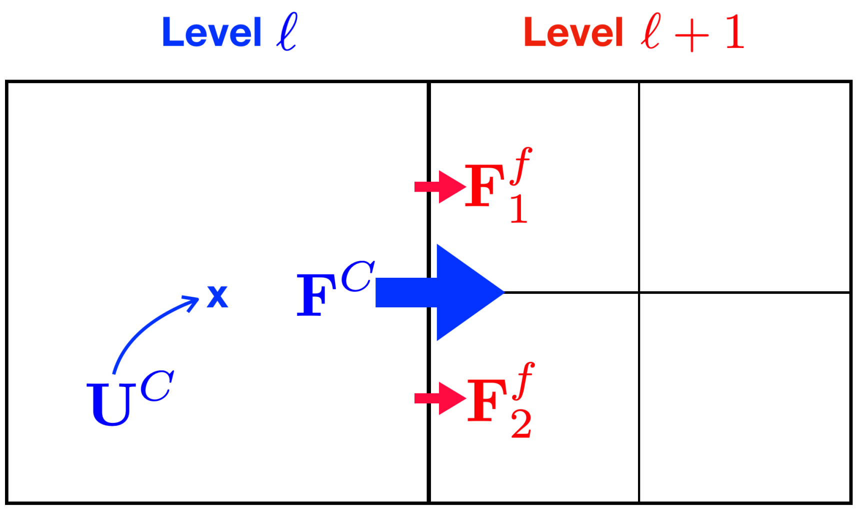


Figure 2.2: Schematic visualization of the AMR refluxing operation for conservation at fine-coarse interfaces. After each Godunov update of the cell-centered conservative fluid variables $\mathbf{U}^C$ on the left coarse cell using the coarse fluxes including the flux $\mathbf{F}^C$ that shares the fine cell boundaries, $\mathbf{U}^C$ is to be corrected using the two fine fluxes of $\mathbf{F}_1^f$ and $\mathbf{F}_2^f$.

computed on the coarse faces based on the underlying conservation law. After the update, $\mathbf{U}^C$ is to be corrected by taking out the coarse face flux $\mathbf{F}^C$ and replacing it with the average of the two fine face fluxes, $(\mathbf{F}_1^f + \mathbf{F}_2^f)/2$. In general, this can be done in terms of making a correction to the quantity $\prod \Delta x_d \mathbf{U}^C$ given as

$$\prod_d \Delta x_d \mathbf{U}^C \leftarrow \prod_d \Delta x_d \mathbf{U}^C - \Delta t^C A^C \mathbf{F}^C + \sum_i \Delta t^f A_i^f \mathbf{F}_i^f, \tag{2.2}$$

where the arrow represents the replacement operation. Here, respectively, A^C and A^f are the areas of the coarse and fine cell faces, $\mathbf{F}^C$ and $\mathbf{F}_i^f$ are the coarse and fine fluxes, and Δt^C and Δt^f are the time steps at those levels. The flux is corrected in both time and space due to the subcycling algorithm utilized in AMReX. For other AMR operations related to AMReX, interested readers are encouraged to refer to [225, 228, 224].

2.0.2 GP for Pointwise AMR Prolongation

In this section we introduce the first GP-AMR prolongation method that is suitable for AMR applications where the state data is comprised of pointwise values. In this case the data the GP-AMR model samples are given as pointwise quantities. Let Δx_d denote the distance between points in a **coarse** level in each $d = x, y, z$ direction. Using the posterior mean function in Eq. (1.4), we first devise a pointwise prolongation scheme for AMR, i.e., AMR prolongation of pointwise data from coarse to fine levels. The choice of $\mathbf{x}_*$ will depend on the *refinement ratio* $\mathbf{r} = [r_x, r_y, r_z]$ and there will be $\prod_d r_d$ new points generated for the new level in general. For example, if a refinement by 2 in all three directions in 3D was requested, 8 new points would be generated. To illustrate the process, suppose a $2\times$ refinement was required in 1D. In this refinement two refined data values are to be newly generated for each and every coarse value. Assume here that we utilize a stencil with the GP radius of one (i.e., $R = 1$) in which case the local 3-point GP stencil $\mathbf{f}_i$ centered at each i-th cell for interpolation is laid out as

$$\mathbf{f}_i = [q_{i-1}, q_i, q_{i+1}]^T.$$

In this example, the fine values $q_{i\pm 1/2}$ are generated. Using this stencil, and Equation 1.4 the refined values follow the equation,

$$q_{i\pm\frac{1}{2}} = \mathbf{k}^T_{i\pm\frac{1}{2}} \mathbf{K}^{-1} \cdot \mathbf{f}. \tag{2.3}$$

In 2D or 3D, data values on a standard $(2N+1)$-point stencil are to be reshaped into a 1D local array $\mathbf{f}_s$ in an orderly fashion, where each $\mathbf{f}_s$ includes corresponding multidimensional data reordered in 1D between $s - N$ and $s + N$. This strategy will be fully described in Section 2.1.

A common practice with GP Modeling is to assume a zero prior mean as was done in Equation (2.3). In these implementations this assumption is also used. Something to note is that the GP weights (the vector $\mathbf{k}_*^T\mathbf{K}^{-1}$) are independent of the samples $\mathbf{f}$, and are constructed based on the length scale parameter ℓ and the location of the samples, $\mathbf{x_n}$, or prediction point, $\mathbf{x}_*$, alone. This is particularly useful in block structured AMR applications, as these weights can computed for each level a priori. If the min and max levels are prescribed for each run each level's model weights can be constructed at the beginning of the simulation.

Since the matrix $\mathbf{K}$ is symmetric and positive-definite, we can use the Cholesky solver to help compute $\mathbf{K^{-1}}$, twice faster than the usual LU decomposition. In practice, we compute and save $\mathbf{w}_* = \mathbf{k_*^T K^{-1}}$ using Cholesky followed by back-substituion only once either at an initial grid configuration step or at the first time an AMR level is newly used. In this way, there is never the need to store $\mathbf{K^{-1}}$ and the computational cost of the prolongation is reduced by performing only a dot product between $\mathbf{w}$ and $\mathbf{f}$, instead of a matrix-vector product $\mathbf{k_*^T K^{-1}}$ followed by another dot product with $\mathbf{f}$ for every prolonged point. As a consequence a compact form is revealed,

$$q_{s\pm\frac{1}{2}} = \mathbf{w}_{s\pm\frac{1}{2}}\mathbf{f}_s, \quad s = i-1, i, i+1. \tag{2.4}$$

There are many choices available for the covariance function to build the GP kernel. One of the most widely used kernels in Gaussian Process modeling is the squared-exponential (SE) covariance kernel function,

$$K(\mathbf{x},\mathbf{y}) \equiv \Sigma^2 \exp\left[-\frac{|\mathbf{x}-\mathbf{y}|^2}{2\ell^2}\right]. \tag{2.5}$$

The prior mean function is often depicted as a constant mean function for simplicity, i.e., $\bar{f}(\mathbf{x}) = f_0\mathbf{1}$, where $\mathbf{1}$ is a vector whose order is the same as $\mathbf{x}$ and whose entries are all unity. The GP-SE model features three hyperparameters, namely f_0, Σ^2, and ℓ. As previously stated $f_0 = 0$ is stipulated in this chapter. In GP modeling the hyperparameter Σ is used in the posterior covariance function, which is used to assess the "quality" of the GP model constructed based on the sampled data. Since uncertainty is not considered in this application, $\Sigma^2 = 1$ is set as it does not effect the calculation of the posterior mean function. The model using the SE kernel in Eq. (2.5) and Eq. (2.3) with the prescribed hyper-parameter choices is our first formula for the pointwise AMR prolongation.

2.0.3 A GP Prolongation for Cell-Averaged Quantities

For the majority of AMReX and fluid-dynamics application codes, the state data is cell-averaged, as per the formulation of Finite-Volume Methods. The above GP-prolongation for pointwise data has no safe guards for mass conservation. To retain conservation alterations to the covariance kernel function in Eq. 2.5 is made to reflect cell-averaged data. Let $\mathbf{G}$ be the vector of cell-averaged samples, whose elements are $G_i = \langle f(\mathbf{x}_i)\rangle = \frac{1}{\mathcal{V}}\int_{I_i} f(\mathbf{x}_i)d\mathcal{V}$, where $I_i \subset \mathbb{R}^D$ is the D-dimensional cell in which $\mathbf{x}_i$ is the center and Δx_d is the cell length in each d-direction, and $\mathcal{V}$ is the cell volume of I_i. In order to calculate the covariance between cell averaged quantities an integrated covariance kernel as described in [158] is utilized. That is,

$$\begin{aligned} C_{k,h} &= \mathbb{E}[(G_k - \bar{G}_k)(G_h - \bar{G}_h)] \\ &= \int \mathbb{E}[(f(\mathbf{x}) - \bar{f}(\mathbf{x}))(f(\mathbf{y}) - \bar{f}(\mathbf{y}))]dg_k(\mathbf{x})dg_h(\mathbf{y}) \\ &= \iint K(\mathbf{x},\mathbf{y})dg_k(\mathbf{x})dg_h(\mathbf{y}) \end{aligned} \tag{2.6}$$

where

$$dg_j(\mathbf{x}) = \begin{cases} d\mathbf{x} \prod_{d=0}^{D-1} \frac{1}{\Delta x_d} & \text{if } \mathbf{x} \in I_j \\ 0 & \text{else} \end{cases} \tag{2.7}$$

With the use of the squared-exponential kernel as K, Equation (2.6) becomes

$$\begin{aligned} C_{kh} = \prod_{d=0}^{D-1} \sqrt{\pi} \left(\frac{\ell}{\Delta x_d}\right)^2 \Bigg\{ & \left(\frac{\Delta_{kh}+1}{\sqrt{2}\ell/\Delta x_d} \text{erf}\left[\frac{\Delta_{kh}+1}{\sqrt{2}\ell/\Delta x_d}\right] + \frac{\Delta_{kh}-1}{\sqrt{2}\ell/\Delta x_d} \text{erf}\left[\frac{\Delta_{kh}-1}{\sqrt{2}\ell/\Delta x_d}\right]\right) \\ & + \frac{1}{\sqrt{\pi}} \left(\exp\left[-\left(\frac{\Delta_{kh}+1}{\sqrt{2}\ell/\Delta x_d}\right)^2\right] + \exp\left[-\left(\frac{\Delta_{kh}-1}{\sqrt{2}\ell/\Delta x_d}\right)^2\right]\right) \\ & -2 \left(\frac{\Delta_{kh}}{\sqrt{2}\ell/\Delta x_d} \text{erf}\left[\frac{\Delta_{kh}}{\sqrt{2}\ell/\Delta x_d}\right] + \frac{1}{\sqrt{\pi}} \exp\left[-\left(\frac{\Delta_{kh}}{\sqrt{2}\ell/\Delta x_d}\right)^2\right]\right) \Bigg\} \end{aligned} \tag{2.8}$$

Here $\Delta_{kh} = \dfrac{x_{d,h} - x_{d,k}}{\Delta x_d}$.

Starting with the SE kernel we integrate over the two cells, one being the sampled stencil I_k, and the other being the target cell I_*, to get the new GP weight vector,

$$T_{k*} \equiv T(\mathbf{x}, \mathbf{x}_*) = \int_{I_k} \int_{I_*} K(\mathbf{x}, \mathbf{x}_*) dg_k(\mathbf{x}) dg_*(\mathbf{x}_*),$$

where

$$I_* = \bigtimes_{d=x,y,z}^{D} \left[x_{*,d} - \frac{\Delta x_d}{2r_d}, \ \ x_{*,d} + \frac{\Delta x_d}{2r_d} \right],$$

in which $\times$ denotes the Cartesian production on sets. Using the SE kernel, a closed form for T_{k*} is derived,

$$T_{k*} = \pi^{D/2} \prod_{d=x,y,z}^{D} r_d \left(\frac{\ell}{\Delta x_d}\right)^2 \sum_{\alpha=1}^{4} (-1)^\alpha \left[\phi_{\alpha,d} \text{erf}(\phi_{\alpha,d}) + \frac{1}{\sqrt{\pi}} \exp(-\phi_{\alpha,d}^2) \right], \tag{2.9}$$

where for each $\alpha = 1, \dots, 4$,

$$\phi_{\alpha,d} = \frac{1}{\sqrt{2}\ell/\Delta x_d}\left(\Delta_{k,*} + \frac{r_d - 1}{2r_d},\ \Delta_{k,*} + \frac{r_d + 1}{2r_d},\ \Delta_{k,*} - \frac{r_d - 1}{2r_d},\ \Delta_{k,*} - \frac{r_d + 1}{2r_d}\right).$$

Therefore, with the combination of the cell-averaged kernel in Eq. (2.8) and the weight vector in Eq. (2.9), the second GP-AMR formula is obtained given in the integral analog of Eq. (1.4) for cell-averaged data prolongation from coarse to fine levels,

$$\langle f(\mathbf{x}_*)\rangle = \mathbf{T}_*^T\mathbf{C}^{-1}\mathbf{G}, \tag{2.10}$$

where the zero mean is used as before. The vector $\mathbf{G}$ of cell-averaged samples within the GP radius R is given as $\mathbf{G} = [G_{i-R}, \dots, G_{i+R}]^T$. Analogous to the pointwise method, $\mathbf{T}_*^T\mathbf{C}^{-1}$ is cast into a new GP weight vector $\mathbf{z}_*$ to rewrite Eq. (2.10) as

$$\langle f(\mathbf{x}_*)\rangle = \mathbf{z}_*\mathbf{G}. \tag{2.11}$$

Many methods perform interpolation in a dimension-by-dimension manner. In contrast, the above two GP-AMR methods are inherently multidimensional. Moreover, the use of the SE kernel as a base in each d-direction does facilitate to obtain the analytic multidimensional form in Eq. (2.8). Our GP-AMR methods, therefore, provide a unique framework where all interpolation procedures in AMR grid hierarchies are genuinely developed to support multidimensionality. Furthermore, it is worth to point out that the two prolongation schemes in Eqs. (2.4) and (2.11) are merely a straightforward calculation of dot products between the GP weight vectors and the grid data. This is the novelty of the use of GP modeling in AMR prolongation, which reveals two new compact prolongation methods that are surprisingly more simple, not only than one might expect, but also than conventional prolongation methods based on polynomials.

2.0.4 Nonlinear Multi-substencil Method of GP-WENO for Non-Smooth Data

Both of the above GP modeling techniques can suffer from non-physical oscillations near discontinuities. The SE and integrated SE kernels work very well for continuous data, but a type of “limiting” mechanism needs to be implement in order to protect against any type of unphysical oscillations in flow regions with sharp gradients. To address this issue, a method to combine GP models trained on multiple sample locations based on the WENO-JS method is utilized as used in [158, 160]. As will be shown in details below, this GP-WENO approach trains multiple sample locations by calculating nonlinear weights ω_m based on the new GP-based smoothness indicators β_m. These β_m are constructed in terms of measuring the likelihood of the local data on substencils $\mathbf{f}_m$ in accordance with the smoothness of the GP model represented by the GP kernel function $\mathbf{K}_{m,\sigma}$ (or $\mathbf{C}_{m,\sigma}$). There two differences between $\mathbf{K}$ and $\mathbf{K}_{m,\sigma}$ in that (i) $\mathbf{K} \in \mathbb{R}^{M\times M}$ and $\mathbf{K}_{m,\sigma} \in \mathbb{R}^{(2D+1)\times(2D+1)}$, where $M = 2D^2 + 2D + 1$ for each spatial dimension $D = 1, 2, 3$, and (ii) the scale-length hyperparameter for $\mathbf{K}_{m,\sigma}$, σ is a much smaller length scale in accordance with the narrow shock-width spread over a couple of grid spacing. The same differences hold between $\mathbf{C}$ and $\mathbf{C}_{m,\sigma}$ as well.

The first step in this multi-substencil method of GP-WENO is to build $2D+1$ substencil data on each substencil S_m, $m = 1, \ldots, 2D+1$. The data are combined using linear weights γ_m derived from an over-determined linear system relating the weights generated by building a GP model on all substencils S_m and the weights generated from a GP model on a total stencil S. The last step is to take the linear weights γ_m to define nonlinear weights ω_m using the GP-based smoothness indicators β_m [158, 160].

In general, given a total stencil S, choose $2D+1$ sub-stencils, $S_m, m = 0, \ldots 2D$, such that $\bigcap_{m=0}^{2D} S_m = \{\mathbf{x}_{i,j}\}$ and $\bigcup_{m=0}^{2D} S_m = S$, the total stencil. That is, the prolongation will have the form:

$$f_* = \sum_{m=0}^{2D} \omega_m \mathbf{w}_m^T \mathbf{f}_m \tag{2.12}$$

where $\mathbf{w}_m^T = \mathbf{T}_{*,m}\mathbf{C}_m^{-1}$ for the cell averaged prolongation or $\mathbf{w}_m^T = \mathbf{k}_{*,m}^T\mathbf{K}_m^{-1}$ for the point-wise prolongation. The coefficients ω_m are defined as in the WENO-JS method [183],

$$\omega_m = \frac{\tilde{\omega}_m}{\sum_s \tilde{\omega}_s} \quad \text{where} \quad \tilde{\omega}_m = \frac{\gamma_m}{(\epsilon + \beta_m)^p}.$$

For this algorithm $\epsilon = 10^{-36}$ and $p = 2$ are chosen. The terms β_m are the likelihood estimates for the stencil $\mathbf{f}_m$, that is

$$\beta_m = \mathbf{f}_m^T \mathbf{K}_\sigma^{-1} \mathbf{f}_m$$

for the pointwise prolongation and

$$\beta_m = \mathbf{f}_m^T \mathbf{C}_\sigma^{-1} \mathbf{f}_m$$

for the cell averaged prolongation. Notice that, due to the properties of the kernel matrices [158], this can cast as

$$\beta_m = \sum_{i=1}^{2D+1} \frac{1}{\lambda_i} \left(\mathbf{v}_i^T \mathbf{f}_m\right)^2, \tag{2.13}$$

where $\mathbf{v}_i$ and λ_i are the eigenvectors and eigenvalues of the covariance kernel matrix, $\mathbf{K}_{m,\sigma}$ or $\mathbf{C}_{m,\sigma}$. As described in [158, 160], the GP-based smoothness indicators β_m defined in this way is derived by taking the negative log of the GP likelihood of Eq. (1.2). This gives rise to the statistical interpretation of β_m which relates that if there is a shock or discontinuity in one of the substencils, say S_k, such a short length-scale (or rapid) change on S_k makes $\mathbf{f}_k$ unlikely. In other words, the GP model whose smoothness is represented by the smoothness property of its covariance kernel, $\mathbf{K}_{m,\sigma}$ or $\mathbf{C}_{m,\sigma}$, gives a low probability to $\mathbf{f}_k$, in which case β_k – given as the negative log likelihood of $\mathbf{f}_k$ – becomes relatively larger than the other β_m, $m \neq k$.

In this method GP modeling is used both in a regression (prolongation) and in a classification sense. The regression aspect enables prolongation of GP sampled over the longer length-scale ℓ On the other hand, the classification aspect is a shock detection model for the handling of discontinuities. This is employed by examining the likelihood of a GP model with a much shorter length-scale σ, which is then integrated into the eigensystem in Equation (2.13). Smaller than ℓ, the parameter σ is chosen to reflect the short width of shocks and discontinuities in numerical simulations, which is typically over a couple of grid spacings. In this manner, two length scale parameters are used, ℓ for the interpolation model, and σ for shock-capturing detection model.

Another key factor are the weights γ_m. The vector $\boldsymbol{\gamma}$, is the least squares approximation for the weights to construct the optimal combination of the each GP model (constructed using sample data $\mathbf{f_m}$). These weights are retrieved by solving the overdetermined linear system:

$$\mathbf{M}\boldsymbol{\gamma} = \mathbf{w}_* \tag{2.14}$$

where the n-th column of $\mathbf{M}$ is given by $\mathbf{w}_n$, and $\mathbf{w}_*$ is the model weights for the interpolation point $\mathbf{x}_*$ relative to the total stencil S. As mentioned previously, these weights are generated using the length scale parameter ℓ. Note that $\mathbf{M}$ is a potentially sparse matrix, and is constructed from the model weights built on each S_m.

To illustrate this concept an example is explored. Suppose $D = 2$, in which case and the total stencil S is in the 5×5 patch of cells centered at (i, j) and contains 13 data points. The total stencil is subdivided into five 5-point substencils S_m, $m = 1, \ldots, 5$. The natural cross-shape substencil is taken for each S_m on each of which GP will approximate function values (i.e., state values of density, pressure, etc.) at 16 new refined locations, i.e., $(i \pm 1/4, j \pm 1/4)$, $(i \pm 1/4, j \pm 3/4)$, $(i \pm 3/4, j \pm 1/4)$, and $(i \pm 3/4, j \pm 3/4)$. For illustrative purposes, choose $\mathbf{x}_{i+1/4,j+1/4}$ as the location for GP to compute function values for prolongation. Explicitly, five 5-point substencils are chosen as,

$$\begin{aligned}
S_1 &= \left[\mathbf{x}_{i,j-1}, \mathbf{x}_{i-1,j}, \mathbf{x}_{i,j}, \mathbf{x}_{i+1,j}, \mathbf{x}_{i,j+1}\right], \\
S_2 &= \left[\mathbf{x}_{i,j-2}, \mathbf{x}_{i-1,j-1}, \mathbf{x}_{i,j-1}, \mathbf{x}_{i+1,j-1}, \mathbf{x}_{i,j}\right], \\
S_3 &= \left[\mathbf{x}_{i+1,j-1}, \mathbf{x}_{i,j}, \mathbf{x}_{i+1,j}, \mathbf{x}_{i+2,j}, \mathbf{x}_{i+1,j+1}\right], \\
S_4 &= \left[\mathbf{x}_{i,j}, \mathbf{x}_{i-1,j+1}, \mathbf{x}_{i,j+1}, \mathbf{x}_{i+1,j+1}, \mathbf{x}_{i,j+2}\right], \\
S_5 &= \left[\mathbf{x}_{i-1,j-1}, \mathbf{x}_{i-2,j}, \mathbf{x}_{i-1,j}, \mathbf{x}_{i,j}, \mathbf{x}_{i-1,j+1}\right].
\end{aligned} \tag{2.15}$$

In this example, the total stencil S is constructed to satisfy $\bigcap_{m=1}^{5} S_m = \{\mathbf{x}_{i,j}\}$ and $\bigcup_{m=1}^{5} S_m = S$, containing 13 data points whose local indices range from $i-2, j-2$ to $i+2, j+2$, excluding the 12 cells in the corner regions. See Fig. 2.3, for a detailed schematic of the multi-substencil approach.

Using these locations a 13×5 over-determined system is built to obtain the

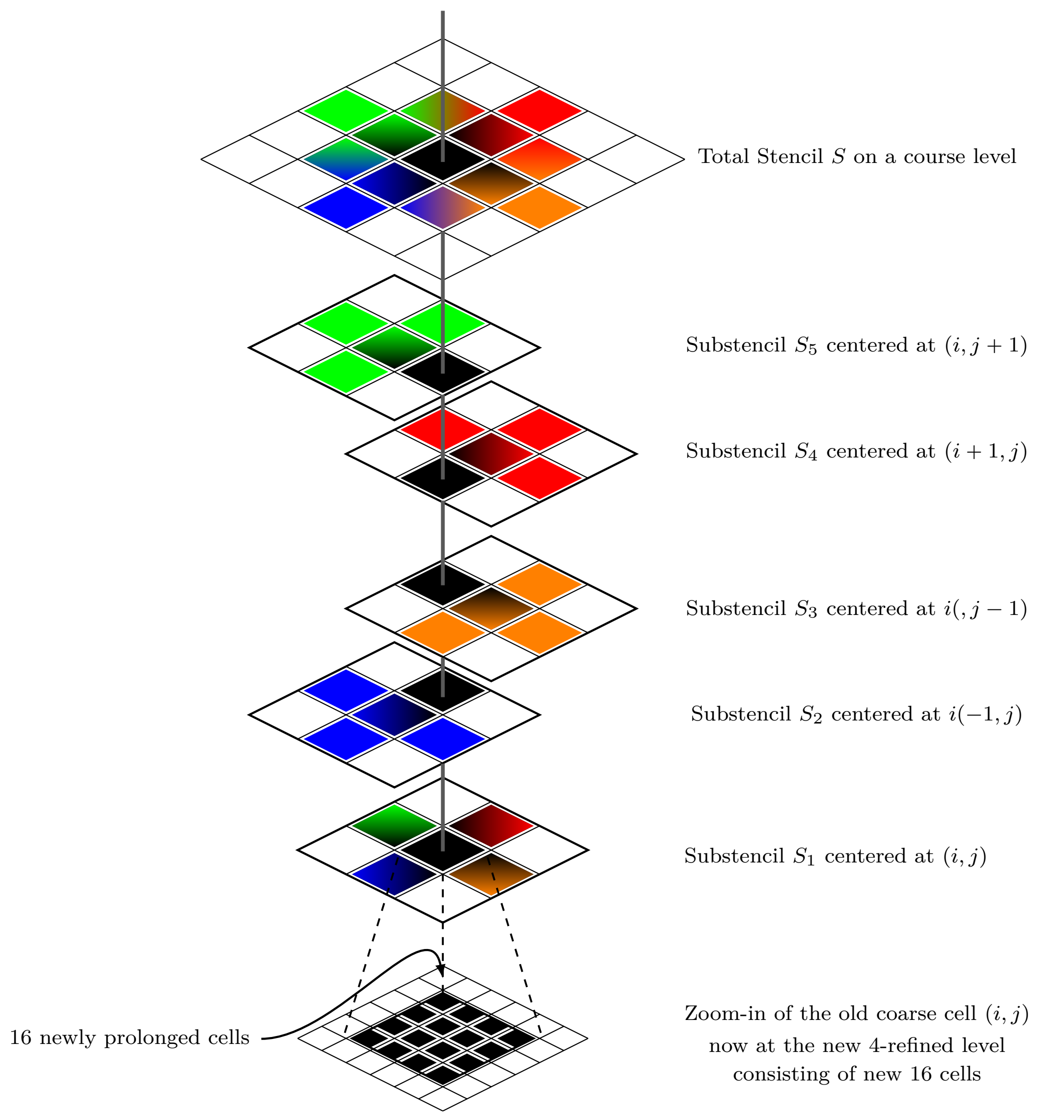

Total Stencil S on a course level
Substencil S_5 centered at $(i, j+1)$
Substencil S_4 centered at $(i+1, j)$
Substencil S_3 centered at $i(, j-1)$
Substencil S_2 centered at $i(-1, j)$
Substencil S_1 centered at (i, j)
16 newly prolonged cells
Zoom-in of the old coarse cell (i, j)
now at the new 4-refined level
consisting of new 16 cells

Figure 2.3: GP prolongation using five GP substencils that are combine to produce 16 new datapoints on a fine 2D grid. The 4× refinement ratio in both x and y directions is considered here to prolong the single data from the old coarse cell (i, j) to 16 newly refined locations.

optimal model combination parameters

$$\begin{pmatrix} w_{0,0} & 0 & 0 & 0 & 0 \\ w_{0,1} & w_{1,0} & 0 & 0 & 0 \\ w_{0,2} & 0 & w_{2,0} & 0 & 0 \\ w_{0,3} & 0 & 0 & w_{3,0} & 0 \\ 0 & w_{1,1} & 0 & 0 & 0 \\ 0 & w_{1,2} & w_{2,1} & 0 & 0 \\ w_{0,4} & w_{1,3} & w_{2,2} & w_{3,1} & w_{4,0} \\ 0 & 0 & w_{2,3} & w_{3,2} & 0 \\ 0 & 0 & 0 & w_{3,3} & 0 \\ 0 & w_{1,4} & 0 & 0 & w_{4.1} \\ 0 & 0 & w_{2,4} & 0 & w_{4,2} \\ 0 & 0 & 0 & w_{3,4} & w_{4,3} \\ 0 & 0 & 0 & 0 & w_{4,4} \end{pmatrix} \begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{pmatrix} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \\ w_{10} \\ w_{11} \\ w_{12} \end{pmatrix}. \tag{2.16}$$

which is solved using the QR factorization method for least squares.

Notice that both SE kernel and the integrated SE kernel in Section 2.0.2 and Section 2.0.3 are both isotropic kernels. Hence, every $\mathbf{K}_{m,\sigma}$ and $\mathbf{C}_{m,\sigma}$ are identical over each substencil, illustrating that the WENO combination weights (i.e., $\mathbf{w}_m^T$) and GP model weights (i.e., $\mathbf{w}_*^T$ and $\mathbf{T}_*^T$) only need to be computed and saved once per level, and reused later.

The nonlinear weighting approach of GP-WENO designed this probabilistic way has proven to be robust and accurate in treating discontinuities [158, 160]. Regardless, the nature of its nonlinearity requires the calculation of nonlinear weights are to be taken place over the entire computational domain, consuming an extra computing time. In this regard, one can save the overall computation if the GP-WENO weighting could only be performed when needed, i.e., near sharp gradients, identified by a shock-detector. In this GP formulation, there is already a good candidate for a shock-detector, that is, the GP-based β_m. To meet this, Eq. (2.13) is slightly modified to introduce an optional switching parameter α, defined by

$$\alpha = \frac{\sum_{i=1}^{2D+1} \frac{1}{\lambda_i} (\mathbf{v}_i^T \mathbf{f})^2}{\mathbb{E}^2_{arith}[\mathbf{f}] + \epsilon_2}. \tag{2.17}$$

Here, the data array $\mathbf{f}$ includes the $2D+1$ data solely chosen from the center most substencil, e.g., S_1 in Fig. 2.3, $\mathbb{E}^2_{arith}$ is the squared arithmetic mean over the sampled coarse grid data points over $2D+1$ sized substencil centered at the cell (i, j, k), or S_1, that is,

$$\mathbf{E}^2_{arith}[\mathbf{f}] = \left(\frac{1}{2D+1} \sum_{\mathbf{x} \in S_1} f(\mathbf{x}) \right)^2, \tag{2.18}$$

and finally, ϵ_2 is a safety parameter in case the substencil data values are all zeros.

Notice that this is just a scaled version of the β_m in eq. (2.13) for the central stencil S_1. Since the GP model with $\Delta\mathbf{x}$ is built with smooth data in mind prescribed by the smooth SE kernel, this parameter will detect "unlikeliness" in of the data $\mathbf{f}$ with respect to the GP model. Note that the critical value of α, called α_c, will be based on the kernel chosen. Without the normalization by the squared arithmetic mean, this factor will vary based on mean value of the data. In this regard, dividing by the average value of the data, $\mathbf{f}$, helps to normalize the factor without changing the variability detection.

It is necessary to choose a critical value, α_c so that shocks, and high variability in $\mathbf{f}$ are detected when $\alpha > \alpha_c$; smooth and low variability when $\alpha \leqslant \alpha_c$. In this dissertation it is heuristically set $\alpha_c = 100$ to meet this strategy. Using this α parameter, there is a switching mechanism between the more expensive multi-modeled GP method and using the single GP model.

Using the multi-substencil GP-WENO method, there are generally $2D + 1$ dot products of the stencil size for each prolonged point, $\prod_d r_d$. In patch-based AMR, even though refined grids are localized around the regions containing shocks and turbulence, there are often areas of smooth flow in every patch. The use of the switch α allows a reduction in the computational complexity to one dot product of the stencil size for each **coarse** stencil that has smooth data, therefore reducing the cost to one dot product of the stencil size for each prolonged point. This method is extremely useful in 3D and when the refinement ratio is greater than $\mathbf{r} = 2$.

This section concludes with a remark on one significant feature of GP which is not explored in this dissertation. The multi-substencil GP-WENO methods, outlined in [158, 160], in smooth flows can variably increase/decrease the order of accuracy. However, in the application for AMR prolongation there may be large grids to be refined, so the increased computational cost can become undesirable. Note that the linear single model GP interpolation is still $\mathcal{O}(\Delta x^3)$, and serves as a high-order accurate prolongation that often matches the order of accuracy of the simulation. Reyes et al. [158, 160] discuss how to vary accuracy as a tunable parameter within the GP methodology. The studies show that the GP radius R of the stencil dictates the order of accuracy. The method illustrated in this paper utilizes a GP radius $R = 1$ and is $\mathcal{O}(\Delta x^3)$, however if one uses $R = 2$ a method that is $\mathcal{O}(\Delta x^5)$ can be retrieved.

2.1 Implementation

This multi-substencil GP method is implemented within the AMReX framework. Due to the complex algorithmic nature of patch-based AMR, and state-of-the-art high performance computing, the AMReX framework is a hybrid C++/Fortran library with many routines. To allow for simple data and workflow, the object-oriented nature of C++ is fully utilized. In AMReX, there is a virtual base class called, *Interpolator*. This class has many derivations including: *CellConservativeLinear*, an object for the functions related to a cell based conservative linear interpolation. The methods provided in this paper live in the *CellGaussian* class. This class constructs a GP object, which contains the model weights for each of the $\prod_{d=0}^{D-1} r_d$ new points per cell as member data. When a simulation is executed in parallelized format, each MPI rank has an Interpolator class, this helps avoid unnecessary communication. Computationally, this is the order of execution:

1. The refinement ratio and $\Delta\mathbf{x}$ are passed on to the construction of the GP object.
2. Build GP covariance matrices for both interpolation $\mathbf{K}, \mathbf{K}_m$ and shock detection $\mathbf{K}_\sigma$
3. Calculate the GP weights for all $\prod_{d=1}^{D} r_d$ prolonged points.
4. Compute the eigensystem of $\mathbf{K}_{m,\sigma}$ as part of building the shock-capturing model.
5. Solve for $\boldsymbol{\gamma}$ for each prolonged point using the weights from S_m and S.
6. For each coarse cell, the switch parameter α is calculated to determine that:
   - the points are prolonged using the nonlinear multi-substencil GP-WENO model (e.g., one of the methods in Sections 2.0.2 and 2.0.3, plus the nonlinear controls in Section 2.0.4) if $\alpha > \alpha_c$, or
   - the linear GP model (e.g., one of the methods in Sections 2.0.2 and 2.0.3 without the method in Section 2.0.4) is used if $\alpha \leqslant \alpha_c$.

If needed, the parameter α_c can be tuned to a different value to alleviate the GP performance relating to sensitivity to shock-detection. By lowering α_c the multi-substencil approach will be utilized more frequently, leading the overall computation to increase since GP-WENO will be activated on an increased number of cells. In most practical applications such a tuning would be unnecessary considering that strong shocks are fairly localized, and in such regions α would retain a value much larger than α_c anyway. Therefore, the condition $\alpha > \alpha_c$ for nonlinear GP-WENO would be met most likely over a wide span of possible values of α_c users might set. Nonethless, the localized nature of shocks allows the computationally efficient linear GP model to be used in simulations that do not require the frequent shock handling mechanism. This value is set $\alpha_c = 100$ in all of the numerical test cases presented in Section 2.2.2.

To illustrate, the α values associated with a Gaussian profile elevated by the circular cylinder of height 0.25 defined by the following function

$$f(x,y) = \begin{cases} 1 + \exp\left(-(x^2+y^2)\right) & \text{if } (x^2+y^2) < 0.5, \\ 0.25 & \text{else.} \end{cases} \tag{2.19}$$

is shown. In Fig. 2.4, it is demonstrated how α varies over the profile which combines the smooth continuous profile with the abrupt discontinuity. It is observed that the α value is close to 2 over the continuous region. However, at the points corresponding to the sharp discontinuity, $(x^2 + y^2) = 0.5$, α soars to over 300, resulting in the full engagement of the multi-substencil GP-WENO model near the discontinuity. In the rest of the smooth region, α becomes much smaller, triggering the linear GP model to be employed effectively. This also shows that the linear GP model would be a sufficient AMR prolongation algorithm in an incompressible setting.

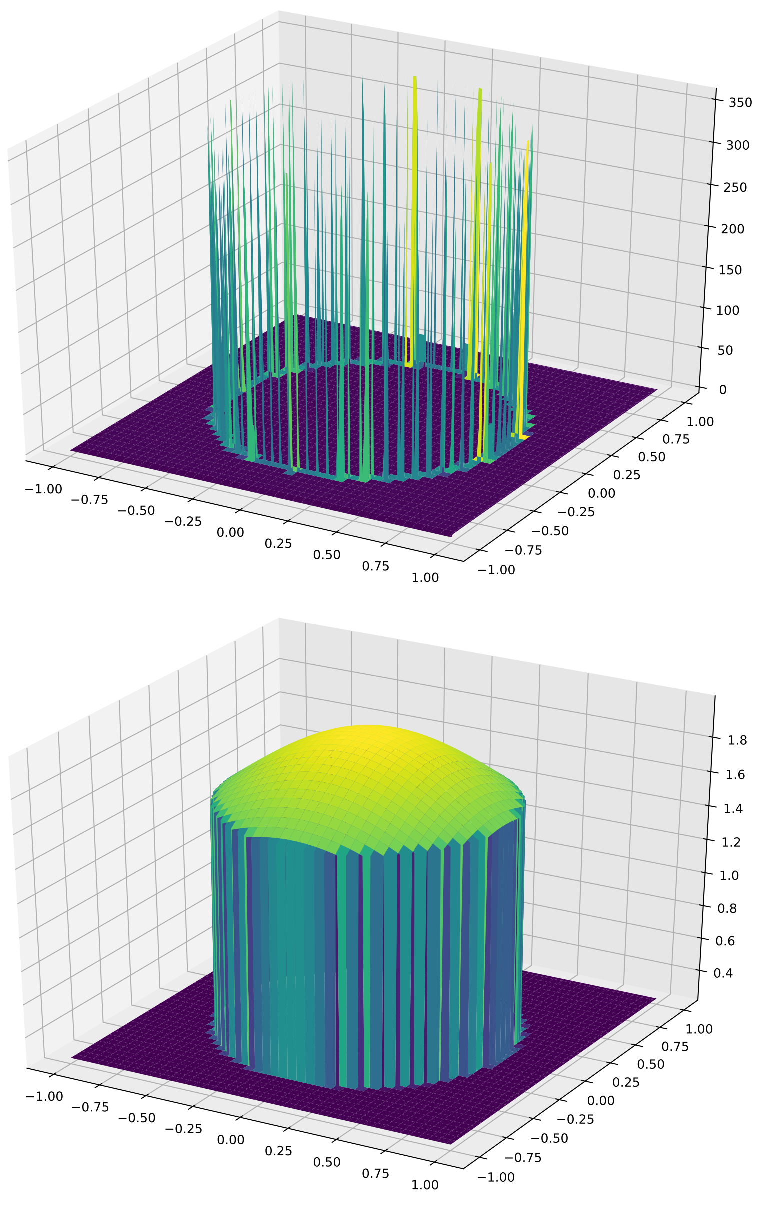


Figure 2.4: The top plot is the α values associated with the data from the function $f(x, y)$, represented by the bottom plot.

2.1.1 AMReX Programming Directives

The implementation is publicly available at `https://github.com/stevenireeves/amrex` in the 'GPAMR' branch. Written in C++, it utilizes AMReX's hardware-agnostic parallelization macros and lambda functions. The code is designed to utilize pragmas that declare the interpolation function as callable from either a CPU or GPU. The AMReX parallelization strategy is similar for both CPU-based supercomputers (e.g., Cori at NERSC) and GPU-based machines (e.g., Summit at the Oak Ridge Leadership Computing Facility (OLCF), as well as Perlmutter at NERSC and the forthcoming Frontier at OLCF). The strategy is to use MPI for domain decomposition, OpenMP for CPU based multi-threading, and CUDA (and HIP in the future) for GPU accelerators. Data allocation, CPU-GPU data transfers and handling are natively embedded in most AMReX data types and objects. For a more in-depth look into how the AMReX software framework is implemented we invite the interested readers to refer to [224].

To provide a simple example into the AMReX style of accelerator programming, suppose that the integer value of 1 needs to be added to a whole AMReX datatype *Array4*. This datatype is a 'plain-old-data' object which is a four dimensional array indexed as (i, j, k, n). The first three indices are for spatial indices and the last one for each individual component (e.g., fluid density, ρ). The AMReX lambda function **AMREX_PARALLEL_FOR_4D** to expand a 4D loop in a parallel fashion is used. For instance, **AMREX_PARALLEL_FOR_4D** distributes the code segment in Listing 1 to an equivalent format in Listing 2:

Listing 2.1: AMReX Directive for Kernel Launch

```
AMREX_PARALLEL_FOR_4D(bx, ncomp, i, j, k, n, {
        my_array(i,j,k,n) += 1;
});
```

Listing 2.2: Expanded For Loop

```
for(int i = lo.x; i < hi.x; ++i){
        for(int j = lo.y; j < hi.y; ++j){
                for(int k = lo.z; k < hi.z; ++k){
                        AMREX_PRAGMA_SIMD
                        for(int n = 0; n < ncomp; ++n)
                                my_array(i,j,k,n) += 1;
                }
        }
}
```

This formulation allows for one code to be compiled for either CPU running or GPU launching. The AMReX lambda functions are expanded by the compiler, and the box dimensions (lo.x - hi.x, etc) are different based on the target device. For GPUs the lo and hi variables are set based on how much data each GPU thread will handle. In the CPU version, the lo to hi are the dimensions of target tile boxes respectively. Essentially the lambda handles the GPU kernel launch or CPU for-loop expansion for the developer/user.

Assuredly, there are other approaches available to launch a parallel region in AMReX for GPU extension. Furthermore an interested reader view the GPU tutorials in AMReX source code for more information on various types of launch macros [224].

While a detailed description of GPU computing is out of scope for this dissertation, if one wishes to implement the GP-AMR algorithm for GPUs a general principle of this strategy is provided:

1. Construct the model weights $\mathbf{T}_*^T\mathbf{C}^{-1}$ for each stencil S_m, $\boldsymbol{\gamma}$ and the eigensystem of $\mathbf{C}_\sigma$ on the CPU at the beginning of program execution or at the initialization of each AMR level.

2. Create a GPU copy for these variables and transfer them to the GPU global

memory space. Every core on the GPU will need to access them, but do not need their own copy.

3. Create a function for the prolongation. This function will require both the coarse grid data as an input, and the fine grid data as an output. Both arrays will need to be on the global GPU memory space. This function will be launched on the GPU, and the fine level will be filled accordingly.

In general, with GPU computing, it is best to do as few memory transfers between the CPU and GPU as possible because a memory transfer can cost hundreds or thousands of compute cycles and can drastically slow down an application. To further explain these steps, Figure 2.5 is of an example call graph along with CPU-GPU memory transfers. In this diagram, it is already assumed that the course and fine state variables have been constructed and allocated on the GPU respectively, as is with the case in AMReX.

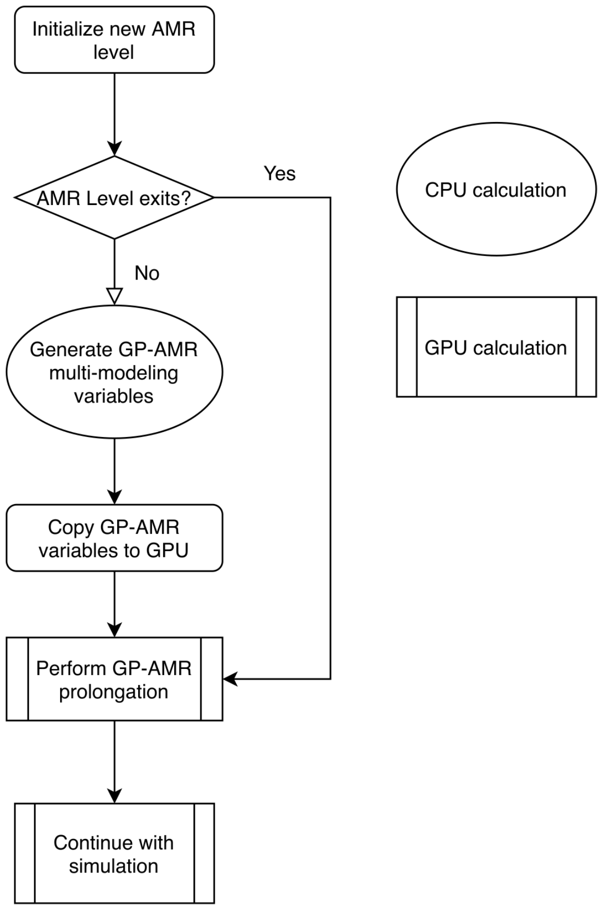


Figure 2.5: Diagram illustrating a call graph for GP-AMR utilizing GPUs as accelerators.

2.2 Results

In this section, the performance of the new GP-based prolongation model is presented in comparison with the default conservative linear polynomial scheme in AMReX. To illustrate the utility of the new GP-based prolongation scheme in fluid dynamics simulations, GP-AMR is integrated as the prolongation method in three different AMReX application codes, including Castro [7] – a massively parallel, AMR, compressible Astrophysics simulation code, PeleC – a compressible combustion code [93], and finally, a simple advection tutorial code built in AMReX.

2.2.1 Accuracy

To test the order of accuracy of the proposed method, a simple Gaussian profile is refined based on the GP prolongation method. This profile follows the formula

$$f(\mathbf{x}) = \exp(-||x||^2) \tag{2.20}$$

where $\mathbf{x} \in [-2, 2] \times [-2, 2]$. The prolonged solution, denoted as f_p, is compared against the true values, f, associated with the Gaussian profile function. The accuracy of the cell-averaged GP prolongation routine matches with the analysis in [158, 160]. The convergence rate of the error in 1-norm, $E = ||f - f_p||$, computed using the GP prolongation model with $R = 1$, exhibits the expected third-order accuracy, following the theoretical slope of third-order convergence in the grid scales, $\mathcal{O}(\Delta x^3)$.

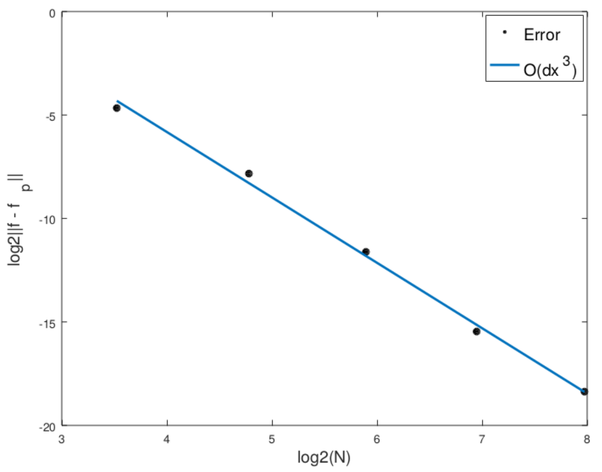


Figure 2.6: Convergence for the GP method. The quantities are measured in log of base 2 to better cope with the refinement jump ratio of 2.

2.2.2 GP-AMR Tests

There are several test problems that are used to demonstrate the capabilities of GP-AMR. First will be a single vortex advection, as per the *Advection_AmrLevel* tutorial in AMReX. Next a modified version of the slotted cylinder problem from [41] is presented. The subsequent problems using Castro are some classic test problems including the Sedov implosion test [176], the double Mach reflection problem [218], and the Kelvin-Helmholtz instability setup [114]. Lastly, a pre-mixed flame simulation from PeleC [93] is illustrated.

The first test is a simple reversible vortex advection run. A radial profile is morphed into a vortex and reversed back into its original shape. The radial profile initially is defined by

$$f(x,y) = 1 + \exp\Big[-100\left((x-0.5)^2 + (y-0.75)^2\right)\Big]. \tag{2.21}$$

The profile is advected with the following velocity field:

$$\mathbf{v}(x, y, t) = \nabla \times \psi \tag{2.22}$$

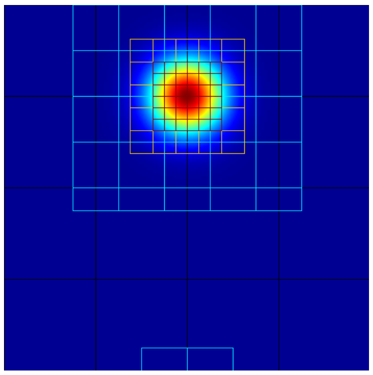
which is the curl of the stream function

$$\psi(x, y, t) = \frac{1}{\pi} \sin(\pi x)^2 \sin(\pi y)^2 \cos\left(\pi \frac{t}{2}\right) \tag{2.23}$$

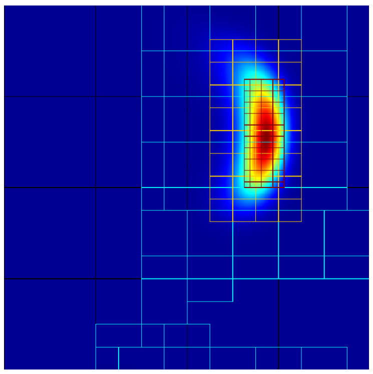
Here $(x, y) \in [0, 1] \times [0, 1]$. In this demonstration, the level 0 grid size is 64×64, and has two additional levels of refinement surrounding the radial profile. The simulation is an incompressible advection problem using the Mac-Projection to compute the incompressibility condition enforcing the divergence-free velocity fields numerically, $\nabla \cdot \mathbf{U} = 0$ [8]. The flux is calculated by a simple second-order accurate upwind linear reconstruction method. Although the overall solution is second-order which is lower than the third-order accuracy of the GP prolongation method, this example still illustrates the computational performance of the GP-based method over the default conservative linear prolongator.

The simulation is finished at $t = 2$. We used sub-cycling of time-steps to improve the overall performance, in which a smaller time-step Δt_f is used on a finer level to advance the regional solutions for stability. The coarser level solutions which advance with a larger time-step Δt_c await until the solutions on the finer levels catch up with the global simulation time $t_g = t^n + \Delta t_c$ over the number of sub-cycling steps $N_{\text{subcycle}} = \Delta t_c / \Delta t_f$. We present the performance and accuracy results for this problem in Table 2.1.

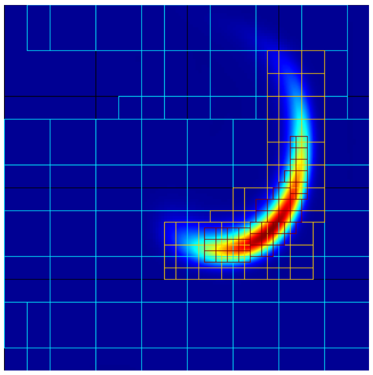
Since the two methods are of different order, they can yield different AMR level patterns which can lead to the slight difference in the number of function calls. In regards to the prolongation functions the default linear prolongation took



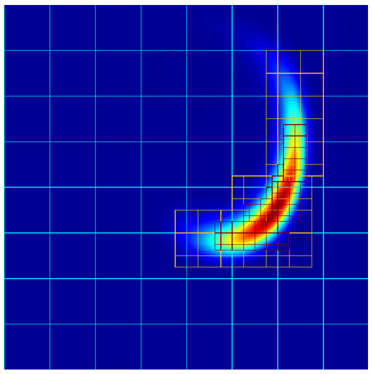
(a) $t = 0$



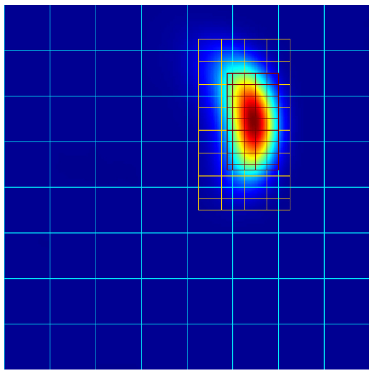
(b) $t = 0.282$



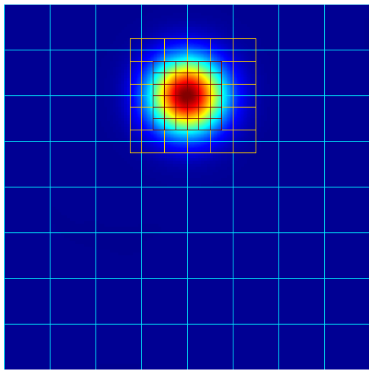
(c) $t = 0.651$



(d) $t = 1.447$



(e) $t = 1.785$



(f) $t = 2$

Figure 2.7: The progression of the 2D radial profile with 4 levels of refinement using the multi-substencil GP prolongation algorithm.

Table 2.1: Accuracy and Performance of GP-AMR against the default linear AMR for the single vortex test on a workstation with an Intel i7-8700K processor, with 6 MPI ranks.

	Execution Time	Prolongation Time	# of calls	L_1 error
$2D$ GP-AMR	0.2323s	0.004168s	9115	0.00033
$2D$ Linear	0.2335s	0.008436s	9113	0.00071
$3D$ GP-AMR	1.6523s	0.086361s	21929	0.00151
$3D$ Linear	1.6640s	0.157623s	21893	0.00160

approximately twice as much time than the GP prolongation. This is due to the smoothness of the solution not requiring the multi-modeled treatment, allowing for the simplified GP algorithm to be used. However, the overall simulation times were equally comparable, since there were larger areas (or more cells) that followed the profile and were computed in the finest AMR level in the GP case than the linear case. We note that the cost of computing the GP model weights were negligible in comparison to the program's execution time of 0.0002306 seconds on average, being called twice (since there were two levels) per MPI rank. We also check the level averaged L_1 error between the solution at $t = 2$ and the solution at $t = 0$ for both AMR prolongation methods. Furthermore, in the 2D case the GP-AMR solution is approximately half of the error produced by the default method.

Another useful examination is the analogous problem in 3D in which the computational stencils for both methods grow. For the 3D version a $32 \times 32 \times 32$ base grid with 2 levels of refinement is used. The details of this simulation can also be found in Table 2.1. Note that a parallel copy operation becomes slightly more expensive with GP because the need for the GP multi-substencil grows on non-smooth regions to handle discontinuities in a stable manner, as managed by the α_c parameter. This becomes more apparent in 3D, as the computational stencil effectively grows from 7 to 25 cells when using the multi-substencil approach. In this 3D benchmark, the difference in error between these methods is less than in the 2D case. The GP-AMR simulation still outperforms the linear prolonged simulation, but to a smaller degree. This is due to the solution at $t = 2$ to be more out of phase with the initial profile with the higher degrees of freedom.

In Figure 2.7 the same 2D single vortex advection with GP-AMR on a base grid of 64×64 with 4 levels of refinement is shown to illustrate the GP method with a more production level grid configuration.

Slotted Cylinder as another AMReX Test

Another useful test is the slotted cylinder advection presented in [41]. In this chapter an exact replica of this problem is not used, but instead the slotted cylinder is put through a similar transformation as in the previous problem. That is, the slotted cylinder is morphed using the same velocity used in the Single Vortex test.

The slotted cylinder is defined as a circle (in 2D) of radius $R = 0.15$ centered at $\mathbf{x}_c = (x_c, y_c) = (0.5, 0.75)$ with a slot of width $W = 0.05$ and height $H = 0.25$ removed from the center of the cylinder. The initial condition is given by

$$\phi_0(\mathbf{x}) = \begin{cases} 0, & \text{if } R < \sqrt{(x-x_c)^2 + (y-y_c)^2}, \\ 0, & \text{if } |2x_c| < W \text{ and } 0 < y_c + R < H, \\ 1, & \text{else}, \end{cases}$$

where $(x, y) \in [0,1] \times [0,1]$. The initial profile is shown in Figure 2.8.

In this test one wants to find the simulation that best retains the profile of the initial condition when it is completed at $t = 2$ as in the previous test. We have two levels of refinement on a base grid of size 64×64 resolution. Figure 2.9 contains snapshots of the simulations at times $t = 0.28, 1.44$ and $t = 2$. The goal is to retain as much of the initial condition as possible, in a similar fashion to the previous 2D vortex advection test.

The result shows that the multi-substencil GP-AMR prolongation preserves the initial condition better than the conservative linear scheme native to AMReX. Notably, there is far less smearing and the circular nature of the cylinder is better retained with GP-AMR. Furthermore, it should be noted that a larger area of

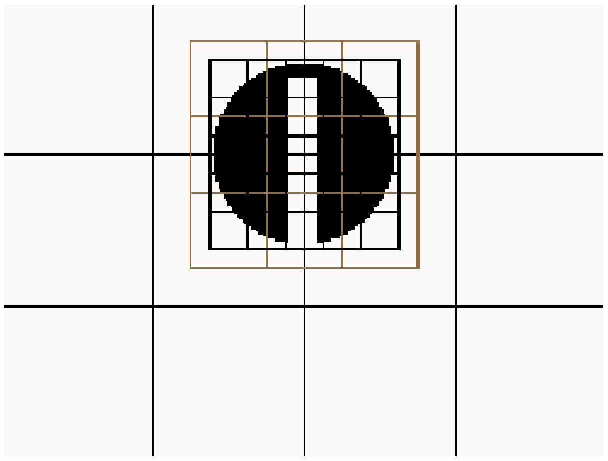
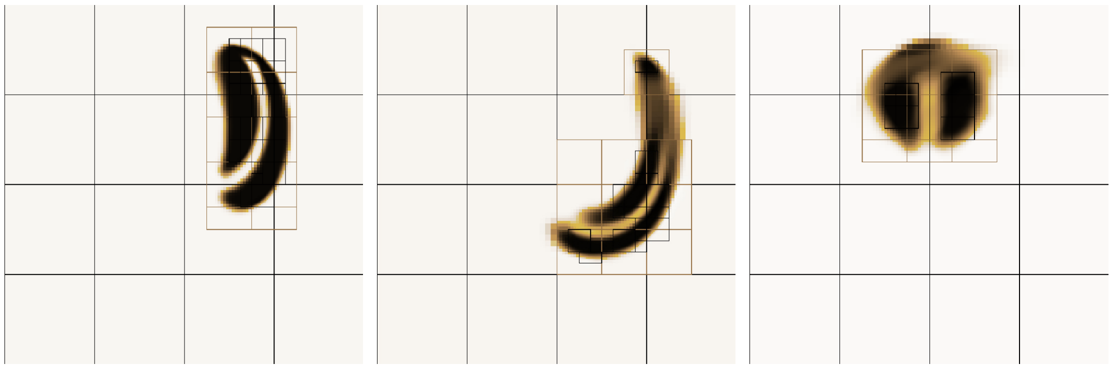


Figure 2.8: The slotted cylinder at $t = 0$ over the entire domain with 3 AMR levels.

the slotted cylinder is covered by the finest grid structure with the GP-AMR prolongation. The refinement criteria in this test and the previous are set for critical values of the profile. This is analogous to refining on regions of high density or pressure. We wish to trace the slotted cylinder's evolution with the finest grid. In this way, a direct comparison of the diffusivity in each method can be seen in how they retain this grid. With this test, the default linear prolongation is much more numerically diffusive and smears the profile almost immediately. This results in a far more blurred cylinder at $t = 2$. While there is some loss with GP-AMR, the profile at $t = 2$ far better resembles the cylinder at the onset of the simulation.

Sedov Blast Wave using Castro

A perhaps more useful test of the algorithm is in a compressible setting, where shock-handling becomes necessary. To illustrate the compressible performance of this method, the Sedov Blast wave [176], a radially expanding pressure wave, is utilized. This simulation is solved using Castro with the choice of the piecewise


(a) Linear AMR prolongation

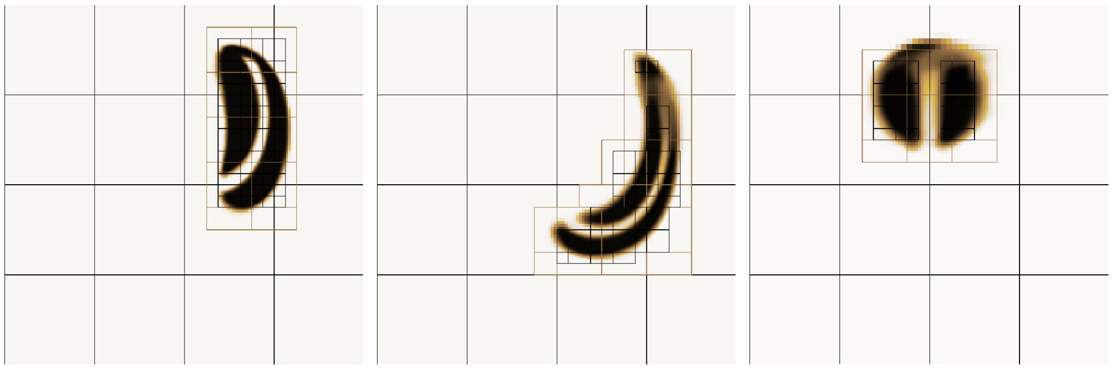
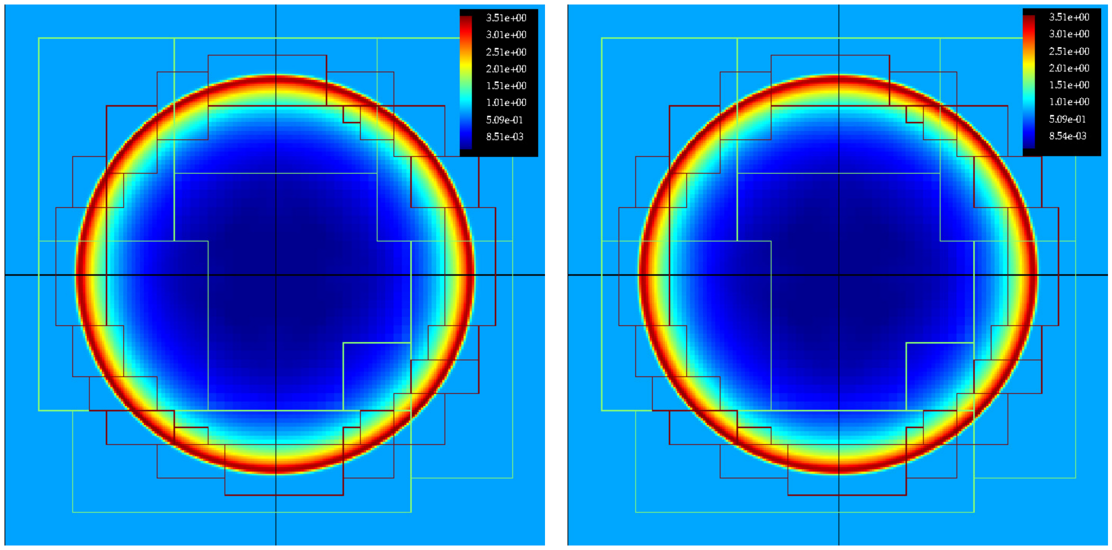

(b) GP-AMR prolongation

Figure 2.9: The morphed slotted cylinder problem at times $t = 0.28, 1.44, 2$, from left to right in time. Top: (a) Default AMReX with linear prolongation. Bottom: (b) AMReX with adaptive multi-modeled GP prolongation.



(a) Sedov with GP-AMR. **(b)** Sedov with linear.

Figure 2.10: A Sedov Blast Wave solution at $t = 0.1$ with two levels of refinement.

parabolic method (PPM) [40] for reconstruction along with the Colella and Glaz Riemann solver [43]. For the 2D test, the simulation has a base grid of 64×64, two additional AMR levels, using a $r_x = r_y = 2$.

Figure 2.10 illustrates the propagation of the Sedov blast wave at $t = 0.1$, and allows a comparison between the linear prolongation method and GP-AMR. Although simple, the Sedov blast wave is a good test illustrating the shock-handling capabilities of the GP multi-substencil model. Notice that visually, the radial shockwaves in both simulations are identical. However, the vacuum in the center of the blast wave is closer to 0 with GP-AMR, is in the self-similar solution [176]. In Figure 2.10 the AMR levels track the shock as it propagates radially and the shock front is contained at the most refined level. At the most refined level, the shock is handled by the multi-modeled GP-WENO treatment. This increases the computational complexity in this region. However, the GP algorithm is less expensive in this example, because the shock is very well localized, and the majority

Table 2.2: Performance of GP-AMR against the default linear AMR on the Sedov Blast Wave with 6 MPI ranks on an Intel i7-8700K processor.

	Execution Time	Prolongation Time	# of calls
$2D$ GP-AMR	5.719s	0.07691s	19743
$2D$ Linear	5.698s	0.12610s	19439
$3D$ GP-AMR	64.27s	1.28912s	35202
$3D$ Non-Adapitve GP-AMR	72.81s	5.09467s	35202
$3D$ Linear	67.76s	1.96204s	35202

of the domain is handled by the regular GP model. The standard GP model is a simple dot-product using the pre-computed weights and the stencil. Table 2.2 contains the performance statistics of the GP-AMR algorithm compared to the default linear using the same workstation as the previous test.

A 3D Sedov blast was also tested, giving a better look at the multi-substencil cost in the shock regions. For this benchmark, the simulation utilized a base grid of $32 \times 32 \times 32$ with additional two levels of AMR, utilizing a refinement factor of 2 for both levels. The wave was advected until $t = 0.01$ with both simulations (GP-AMR and default) 118 coarse grid timesteps. Table 2.2 also contains the performance metrics for the 3D test.

By setting $\alpha_c = 0$ the multi-substencil GP-WENO method is effectively used over all cells. The metrics for this example are labeled as "Non-Adaptive GP-AMR" in Table 2.2. Using the multi-substencil GP-WENO method for every grid is roughly $5\times$ more expensive as a prolongation method. This is expected as the multi-modeled GP-WENO method combines 5 GP models.

Double Mach Reflection using Castro

The double Mach reflection [218] is a great problem for testing purposes due to the complex nature of the solution. For this test, Castro with PPM [40] reconstruction and the HLLC [206] Riemann solver is used. The initial condition

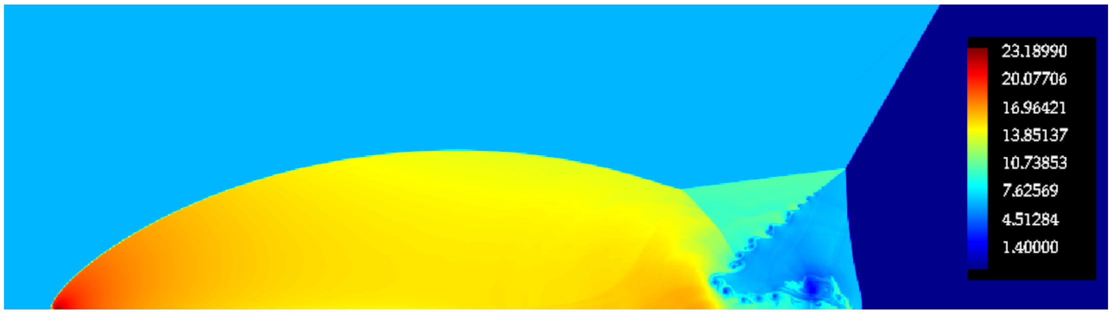


Figure 2.11: The double Mach reflection simulation at $t = 0.2$ with 4 levels of AMR refinement.

describes a planar shock front with an angle of $\theta = \pi/3$ extending from the x-axis which itself is a reflecting wall,

$$\Big(\rho, u, v, p\Big) = \begin{cases} (1.4,\ 0,\ 0,\ 1) & \text{for} \quad x > x_{shock}, \\ (8,\ 8.25,\ -8.25,\ 116.5) & \text{else}, \end{cases} \tag{2.24}$$

where

$$x_{shock} = \frac{y + \frac{1}{6}}{\tan \frac{\pi}{3}}$$

when $y \in [0, 1]$. The full domain of the problem is $[0, 4] \times [0, 1]$.

Figure 2.11 is of the solution to this problem with 4 levels of refinement starting at a base level with resolution 512×128 using the GP-based prolongation.

With sufficient resolution and accuracy, vortices along the primary slip line can be observed, as seen with the copious amount of vortices in Figure 2.11. The number of vortices serves as a general indication of the numerical diffusivity of the method, and a quality of Riemann solver. In this context, in the amount of numerical dissipation of the two different AMR prolongation methods is of interest.

For reference a labeled schematic of the double-mach reflection containing the

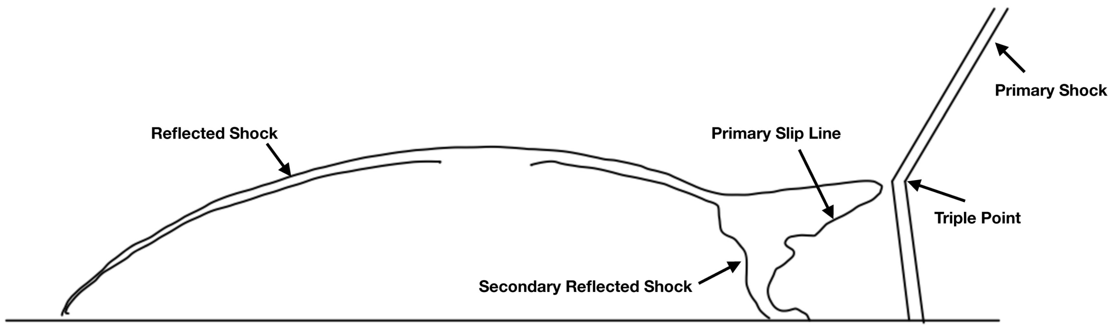


Figure 2.12: A schematic of the main features in the double mach reflection problem.

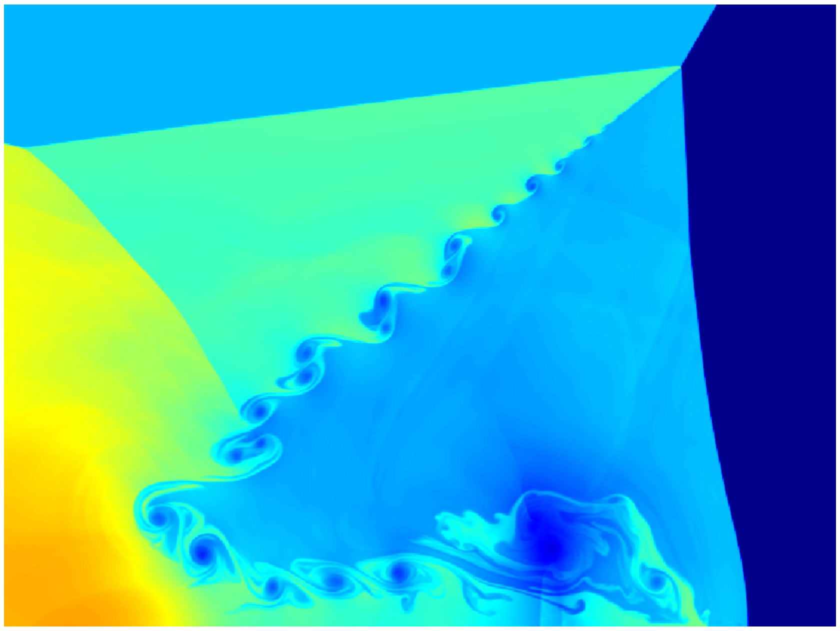
regions of interest for this comparison is presented in Figure 2.12. The features contained in this diagram will be referred to in the following analysis. Mostly in the central region encompassing secondary reflected shock, triple point, and primary slip line.

The default and GP-AMR implementations are compared by zooming into the aforementioned region. In Figure 2.13 the effects of the each prolongation method on the number of vortices along the primary slip line can be observed.

As can be seen in Figures 2.13a and 2.13b, there is more onset to Kelvin-Helmholtz instability along the primary slip line in the GP-based AMR simulation, resulting in more additional vortices above the secondary reflected shock wave, along with onset to instability on the primary slip line close to the primary triple point. As a rudimentary measure, the GP-AMR simulation contains 20 vortices in this region whereas the default AMR contains 17.

With this simulation, the default linear prolongation is faster, as the adaptive GP algorithm has more cells to prolongate in the high α regime. Table 2.3 contains the details about the execution times.

These results were generated using the University of California, Santa Cruz's Lux supercomputer, utilizing 8 nodes. Each node contains two 20-core Intel Xeon


(a)

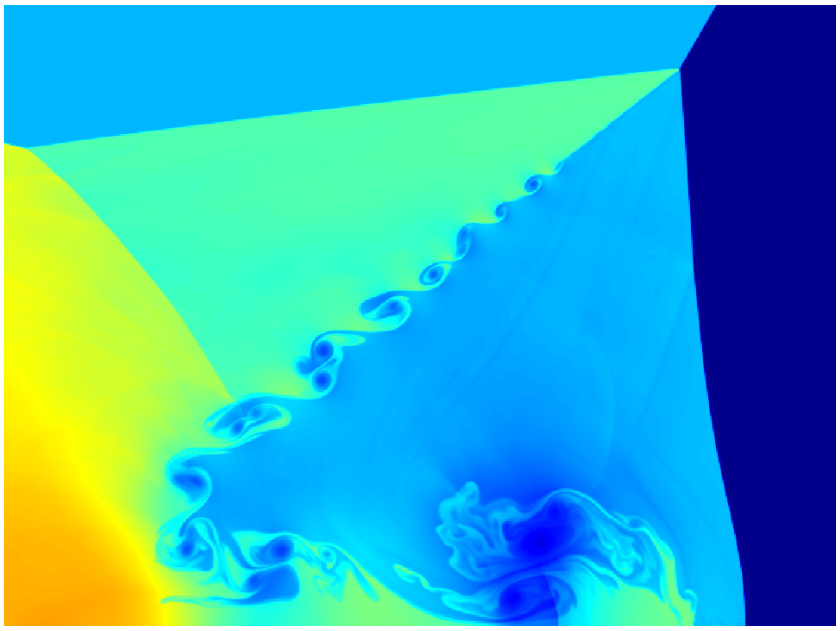

(b)

Figure 2.13: (a) GP-AMR simulation, (b) Default linear prolongation based simulation. Both simulations visualized in the triple-point region of the domain at $t = 0.2$.

Table 2.3: Performance insights for the Double Mach Reflection problem utilizing 8 nodes and 320 cores on Lux.

	Execution Time	Prolongation Time	# of calls
GP-AMR	760.43s	1.1121s	39724
Linear	705.10s	0.7395s	39837

Gold 6248 (Cascade Lake) CPUs. This simulation generates more high α regions, and thus requires the multi-modeled GP-WENO algorithm more often. This results in the GP-AMR simulation to be slower than the default prolongation method by 60s. In addition to the increase of computational complexity, there is an increase of time spent in the parallel copy algorithm. The multi-modeling GP-WENO algorithm requires 2 growth cells at the boarders of each patch, therefore increasing the amount of data to be copied.

Pre-Mixed Flame using PeleC

For the final test problem in this chapter, a steady flame is produced using the AMR compressible combustion simulation code, PeleC [93]. With PeleC, chemical species are tracked as mass fractions that are passively moved during the advection phase, and diffused subject to transport coefficients and evolved in the reaction phase by solving ordinary differential equations to compute reaction rates. In this problem, the GP-AMR is tied with PPM reconstruction and the Colella and Glaz Riemann solver [43]. We show the ρw (momentum in the z direction) of the pre-mixed flame solution in Figure 2.14. For this illustration the base grid had a $32 \times 32 \times 256$ configuration with two additional AMR levels. Furthermore, Figure 2.14 has a colormap such that the lighter color are regions with high momentum, and the dark regions have low momentum. The pre-mixed flame is a 3D flame tube problem in a domain that encompasses $[0, 0.625] \times [0, 0.625] \times [1, 6]$. The flame

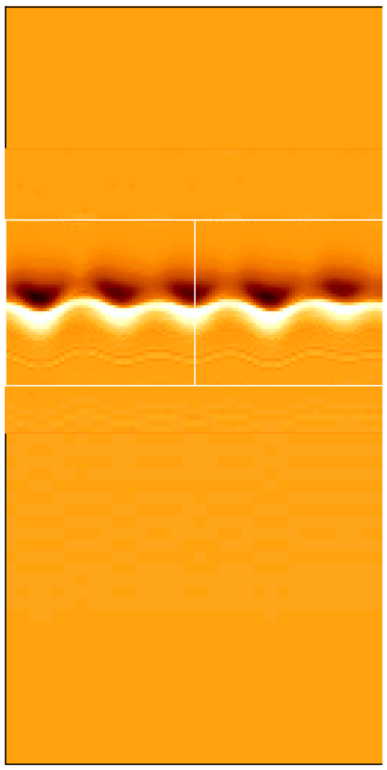
Table 2.4: Execution timings of PeleC and AMReX on the PreMixed Flame test problem on 32 nodes of the Summit supercomputer.

	Execution Time	Prolongation Time	# of calls
GP-AMR	50.23s	0.03281s	940
Linear	52.04s	0.06261s	940

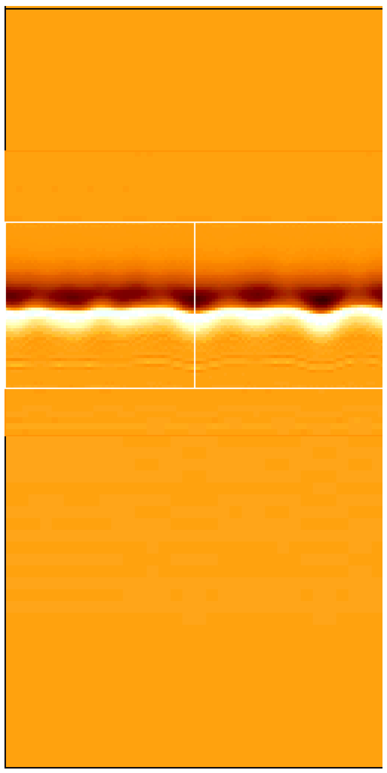
spans the x and y dimension and is centered at $z = 3.0$. The gases are pre-mixed and follow Li-Dryer hydrogen combustion chemical kinetics model [123].

To illustrate some performance metrics the simulation is executed on 32 Summit nodes, with 196 NVIDIA V100 GPUs. The simulation contains a base level of $256\times128\times2048$ with two levels of refinement. The performance metrics of GP-AMR against the default in Table 2.4. In this table GP-AMR is twice as fast as the default linear on average.

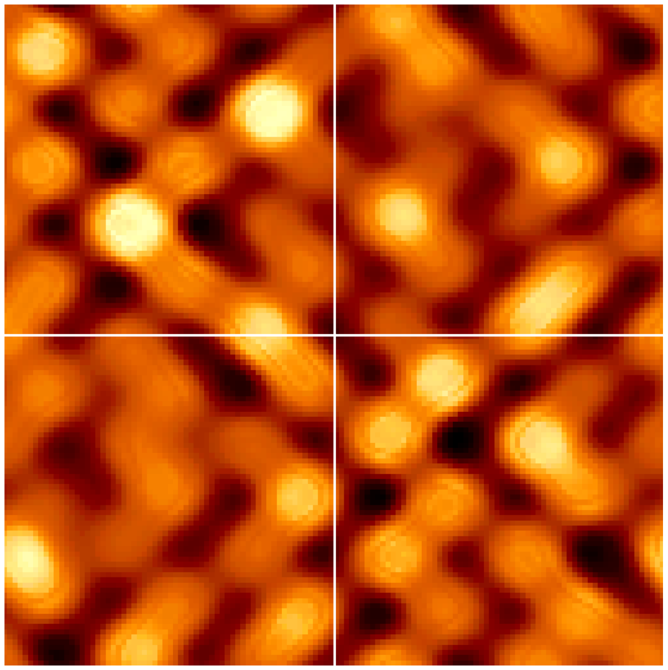
Additionally a weak scaling is performed on this problem for up to 3072 NVIDIA V100 GPUs on Summit, with results illustrated in Figure 2.15. The y-axis of the figure illustrates the average GP prolongation times and the x-axis is number of GPUs from 96 nodes to 3072 GPUs on Summit in logarithmic scale of base-2. Each node on Summit contains 6 NVIDIA V100 GPUs, therefore the scaling ranges from 16 to 512 nodes. GP-AMR when implemented in AMReX scales very well on Summit, one of the top-class supercomputers in the modern leadership computing facilities in the world.



(a) $x - z$ section, $y = 0.3125, z \in [1, 4]$



(b) $y - z$ section, $x = 0.3125, z \in [1, 4]$



(c) $x - y$ section, $z = 3.0$

Figure 2.14: Momentum in the z-direction of the pre-mixed flame.

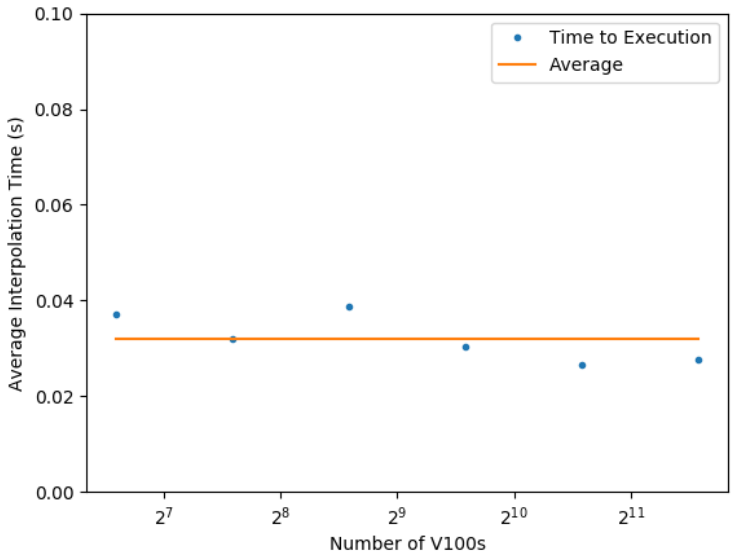


Figure 2.15: Weak scaling of GP-AMR utilized in PeleC up to 3072 Nvidia Volta GPUs (512 Nodes on the OLCF Summit Super Computer).

2.3 Chapter Summary

In this chapter a new, efficient, 3rd order accurate, Adaptive Mesh Refinement prolongation method based on Gaussian Process Regression is presented. Further, this method is general to the type of data being prolonged, as illustrated with a substitution of Covariance kernels in Equations 2.5 and 2.8, for pointwise and cell-averaged data. In order to handle shock waves, a multi-modeled version of the algorithm inspired by WENO [183] was generated. It is recognized that the multi-modeled GP is more computationally expensive, and a method to mitigate this was discovered. By adapting the shock-capturing GP detection model in the multi-modeled GP formulation, a method switching parameter α was discovered.

In the 3 of the 5 test cases, the adaptive GP method was faster than the baseline linear interpolation. The other cases had situations where the patches to be interpolated contained mostly cells where the detection model decided that the multi-modeled approach was necessary. The adaptive method is a balance between speed, stability, and accuracy.

In the scope of this chapter, the tunable parameters ℓ and σ are either fixed, or fixed in relation to the grid scale. To further adapt the algorithm, one could try and maximize the log of Equation 1.2 with respect to the hyper-parameter ℓ as is done in many applications utilizing Gaussian Process regression. However, in this application a fixed prescription for ℓ appears to hold the desired properties without . The stability of the algorithm is inherently tied to the σ parameter, which it is recommended to never being larger than 3 times the grid scale. In many of the test cases, $\sigma = 1.5\Delta x$ was chosen. If additional stability is required, it is recommended to tune α_c to be smaller or to be zero, requiring the algorithm to only use the multi-modeled GP model.

Utilizing the framework provided, an even higher order prolongation method can be generated by just by increasing the size of the stencil while utilizing the same framework. This will be inherently useful as more simulations codes are moving to increasingly accurate solutions with WENO [183, 200] or GP [158, 160] based reconstruction methods paired with Spectral Differed Corrections(SDC) [140, 66] can yield a 4th or higher order accurate total simulation. In this case, a 2nd order AMR interpolation may degrade the overall quality of the solution or incur additional SDC iterations – increasing the execution time of the simulation.

Chapter 3

A GP Algorithm to Upsample Document Images for Optical Character Recognition

An important problem in computer vision is the retrieval of textual information from images of documents. This is especially useful for search engines, accessibility tools for the visually impaired, and for processing of financial documents. For these purposes, optical character recognition (OCR) engines have been constructed. The popular open-source OCR framework Tesseract is used in this study. Optical character recognition frameworks, in general, are only as good as the document image that is supplied to them. In many cases, the resolution of the document image plays a role in how well the characters are extracted. In order to properly upsample low resolution document images, a new Gaussian Process Modeling upsampling algorithm is constructed and presented in this chapter.

This chapter is organized in the following way. To begin, OCR is introduced along with the state-of-the-art OCR extraction software Tesseract. Next, the

Gaussian Process based upsampling method is discussed, along with a brief study on the choice of covariance kernels and the use of a maximum likelihood estimate for the mean. Finally the algorithm is tested against the baseline bicubic upsampling technique by examining the produced OCR accuracy resulting from these upsampled images.

3.1 Optical Character Recognition

Optical Character Recognition is the conversion of pixel represented words and characters within images into machine-encoded text. As previously mentioned, the OCR framework Tesseract [189] is used to extract text in the document images used in this chapter. Tesseract was originally formulated by HP research between 1984 and 1994. Since then it has changed hands and now is an open-source software package managed by Google [147] – under the Apache 2.0 License. Tesseract 4.1.1, the version used in this chapter, generates text based utilizing a type of Neural Network (NN) called a Long-Short Term Memory (LSTM) network. Tesseract ingests single-channel images and generates feature-maps based on these images. Then these feature maps are embedded into an input for the LSTM [189, 147].

A brief on LSTM

To better understand the Tesseract, this section provides a brief introduction into LSTM. A LSTM is a gated Recurrent Neural Network (RNN) that is specifically formulated to avoid a common problem with RNNs, called the vanishing gradient problem. Recurrent Neural Networks typically operate on sequences of data instead of just a single data object alone. Non-LSTM RNNs can keep track of arbitrary long-term dependencies in data sequences. Recurrent Neural Networks are composed of units that “unfold” over the sequence. As such, a RNN generally accepts two inputs, the features at the current time or point in the sequence x_t and the hidden layer or output from the previous unit h_{t-1} or z_{t-1} respectively. During training, where a gradient descent algorithm is utilized, iterative multiplication of dependencies can cause the back-propagated gradient to either diverge or "vanish" (go to zero) [91].

Long Short-Term Memory networks mostly avoid these gradient problems by adding additional complexity to the internals of each network unit. Additionally, an LSTM unit takes 3 inputs: the feature at the current time x_t, the output from the previous unit h_{t-1} and a memory term from the previous unit, c_{t-1}. This memory term informs the current unit on which dependencies to keep. This decision process informs the network whether to forget or keep certain dependencies, and is learned during training [91, 72]. Figure 3.1 contains two units, one from a RNN and one from an LSTM. Each unit contains gates that yield processed results from features. Both the RNN and LSTM contain the sigmoid activation function

$$\sigma(x) = \frac{1}{1 + \exp(-x)}, \tag{3.1}$$

and the LSTM utilizes another activation, $\phi(x) = \tanh(x)$ [146]. In addition to these gates (layers), the LSTM unit contains many point-wise operations on the feature outputs from the gates. Long Short-Term Memory networks have been used to great success in unsegmented-connected handwriting recognition [78], speech recognition [125, 166], and OCR [189]. Tesseract specifically, uses LSTM to generate words from character feature sequences.

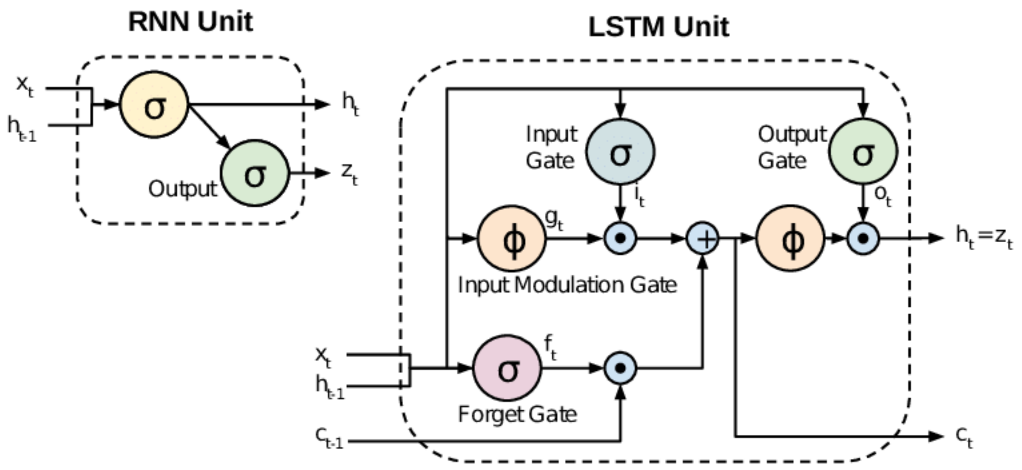


Figure 3.1: An RNN unit vs an LSTM unit [94].

3.2 The Gaussian Process Algorithm

Images are quite different than the type of data presented in Chapter 2. Each entry in an image is typically an 8 bit unsigned integer – between 0 and 255 – in contrast to the 64 bit floating point representations necessary in Computational Fluid Dynamics. They are in general, composed of 3 color channels, and sometimes an α channel. However, for the purposes of OCR, only one channel is considered– a composite gray-scale derived from the RGB channels.

Image data is inherently discontinuous, and does not require the same conservative qualities that the algorithms in Chapter 2 do. To this avail, a different covariance kernel function is used.

Text in single-channel document images are defined by pixels with low intensity values (close to 0 or black), surrounded by pixels of high intensity (closer to 255 or white). Specifically, pixel values are low in the interior of a character, and pixel values are comparatively high outside of characters. Because of this specific structure, the type of GP modeling will change. Instead of modeling the raw values, the deviation from a mean intensity will be modeled. This allows the upsampling algorithm to better maintain these intensities in the presence of characters. This structure is discussed in more detail in subsection 3.2.2 and 3.2.3.

3.2.1 Choice of Covariance Kernel

Due to the nature of the partial differential equations used in CFD, the solutions are continuous almost everywhere. This is why a natural choice of kernel for that application was the squared-exponential kernel – since functions sampled from a Gaussian Process built with the squared-exponential kernel belong to $\mathcal{C}^{\infty}$. However, with images, an alternative kernel is more appropriate for generating data of the same caliber. So instead of the SE kernel, a member of Matérn family of kernels is used. In the Matérn family of covariance functions, there are 3 hyper-parameters that dictate their character – as indicated in Equation (3.2).

$$K_{mat}(\mathbf{x}, \mathbf{y}) = \Sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{||\mathbf{x} - \mathbf{y}||}{\ell} \right)^{\nu} K_{\nu} \left(\sqrt{2\nu} \frac{||\mathbf{x} - \mathbf{y}||}{\ell} \right) \tag{3.2}$$

For the Matérn kernels, Σ and ℓ are the same as they are in the squared-exponential. The hyper-parameter ν on the other hand, relates the level of "continuousness" of the functions that are sampled from a Gaussian Process constructed from this kernel. The function K_{ν} is the modified Bessel function of the second kind of order ν. The Matérn family of covariance functions give continuity properties ranging from functions generated by using Ornstein-Uhlenbeck covariance function, $K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{||\mathbf{x}-\mathbf{y}||}{\ell}\right)$, and the SE covariance function. A Gaussian Process with the Ornstein-Uhlenbeck kernel generates functions that are never differentiable, essentially the opposite of the SE kernel. On one end of the spectrum ,as ν goes to infinity, the Matérn covariance kernel converges to the squared-exponential covariance function. And if $\nu = 1/2$ is selected, the Matérn covariance function simplifies to Ornstein-Uhlenbeck [155]. Note that these are the cases when the distance metric is Euclidean, and the Matérn family holds difference properties on spaces that are formed with other metrics.

Consideration of the input and output datatypes of the Gaussian Process are key when choosing or building a covariance function, as can be seen with the analysis in sections 2.0.2 and 2.0.3. The datatype for this application are document images, which contain sharp contrasts that are handled better by a low ν Matérn kernel. The Matérn kernel with $\nu = 3/2$ is used in this algorithm. For this specific value, Equation (3.2) can be simplified. By setting $\nu = 3/2$,

$$K_{3/2}(\mathbf{x}, \mathbf{y}) = \Sigma^2 \left(1 + \sqrt{3}\frac{||\mathbf{x} - \mathbf{y}||}{\ell}\right) \exp\left(-\sqrt{3}\frac{||\mathbf{x} - \mathbf{y}||}{\ell}\right). \tag{3.3}$$

As in all the algorithms in this dissertation , $\Sigma = 1$ is chosen, as the uncertainty portion of GP modeling will not be used for this application. Figure 3.2 contains a graph with a cross-section of the covariance kernels as a function of distance between two points x and y. The blue curve represents the Squared-Exponential covariance function and the orange curve is the Matérn 3/2. The Matérn 3/2 kernel decays sharper as the the distance increases, and is not as smooth in its transitions like the SE function is. In this illustrative example, $\ell = 1$ is used in both functions.

In order to discuss the practical difference between the Matèrn 3/2 kernel and the Squared Exponential, Figures 3.3 and 3.4 are generated utilizing functions from the Scikit Learn framework [148]. These figures contain prior and posterior mean functions of the Gaussian Processes generated using the aforementioned covariance kernels. The prior mean functions sampled from the Gaussian Processes offer illustrations of typical functions that "live" in the function spaces that the covariance kernels stipulate. Ten data points are selected from a uniform distri-

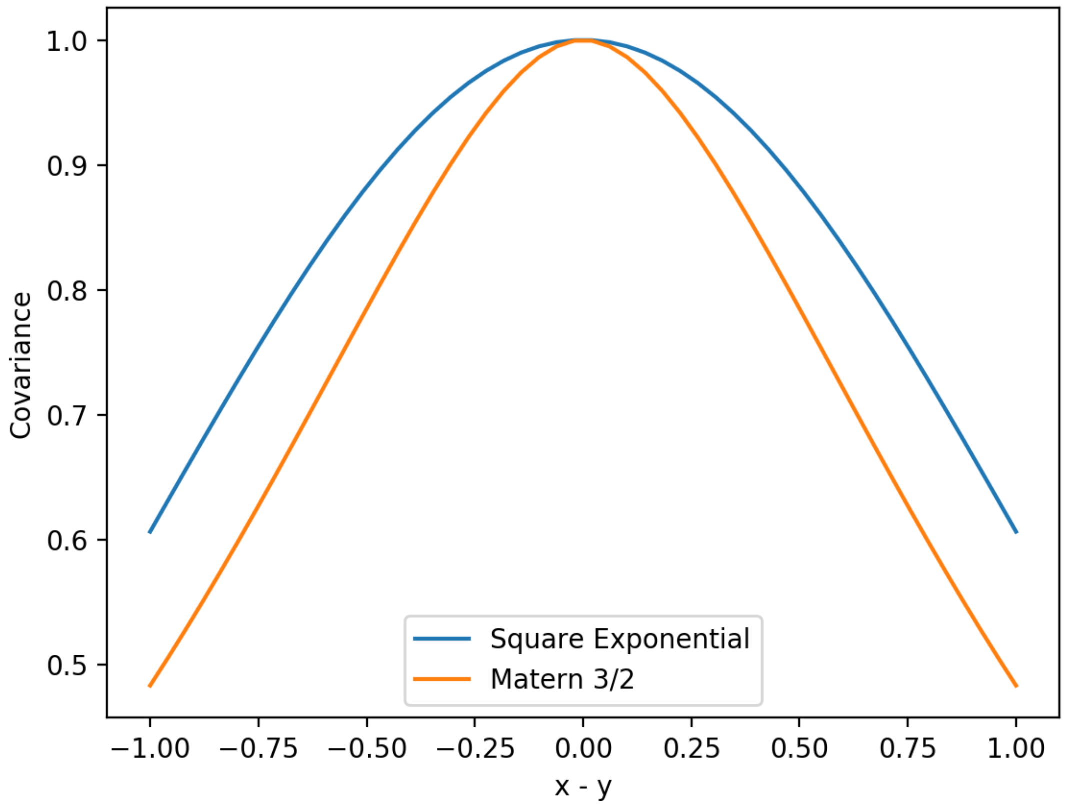


Figure 3.2: Cross-Section of the Squared-Exponential and Matérn 3/2 Covariance Functions.

bution between 0 and 5. The sampled response variable follows the formula

$$Y = \sin\left((X - 2.5)^2\right) \tag{3.4}$$

where $X \sim \mathcal{U}(0, 5)$ is the predictor variable. Figure 3.3 contains the prior and posterior mean functions generated from Gaussian Process with the Squared Exponential Kernel using these datapoints. The gray space represents the uncertainty of the Gaussian Process models. For Figure 3.4 the above process is repeated utilizing the Matèrn 3/2 kernel instead of Squared-Exponential. Note that in Figure 3.3, the prior and posterior mean functions are much smoother than the functions sampled from and produced by the GP with the Matérn kernel, as represented in Figure 3.4. As mentioned previously, the Squared Exponential based GP expects input and output functions to be $\mathcal{C}^{\infty}$, whereas the Matèrn Kernel essentially generates its differentiability properties based on the value of ν [155].

3.2.2 Maximum Likelihood Estimate for the Prior Mean

As with the application for AMR, Equation (1.4) is used as a base for the upsampling algorithm. However, in this chapter, a zero prior mean is not assumed but rather a prior mean function is built. The prior mean function that will be used is the *maximum likelihood estimate for the prior mean*, calculated over the 5×5 square patch of pixels. This is done to change the character of the upsampling model so the model predicts the variation about the mean intensity in each sample. Typically, non-zero mean functions are used when there is an observed or assumed trend in the data. In the case of document images, pixel data is expected to retain certain intensities when inside a character or in the white space of a document. Because of these characteristics, a constant non-zero mean is chosen. Note that the derived prior mean functions is only constant over a single window, the prior

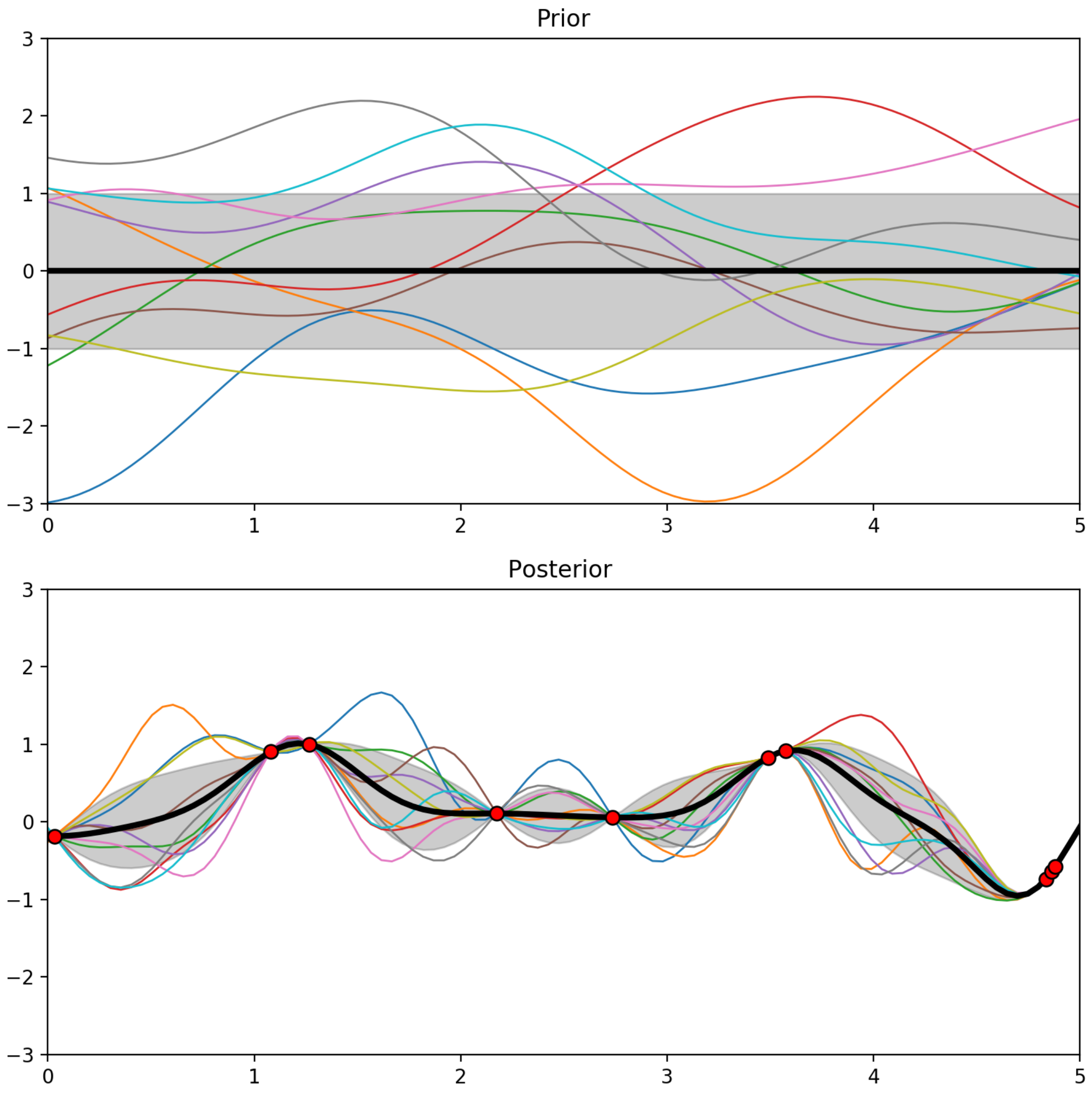


Figure 3.3: Squared Exponential based Gaussian Process, fitted to ten data points. Top: Prior Mean functions. Bottom: Posterior mean functions based on the 10 samples.

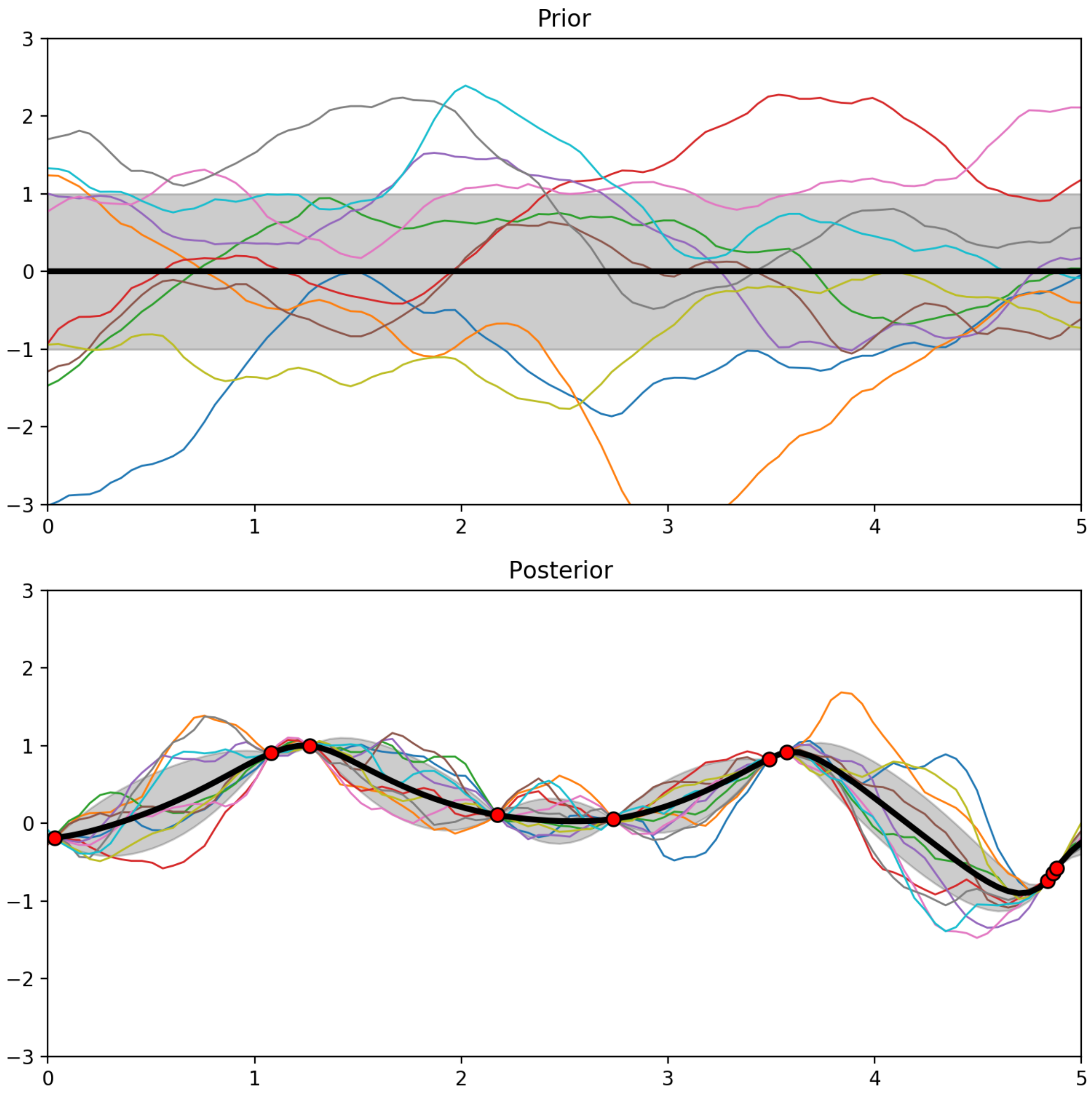


Figure 3.4: Matèrn 3/2 based Gaussian Process, fitted to ten data points. Top: Prior Mean functions. Bottom: Posterior mean functions based on the 10 samples.

mean will be constructed over each sample varies over the image.

To calculate the maximum likelihood estimate (MLE) for a constant prior mean, $\bar{\mathbf{f}} = f_0\mathbf{1}$, the log of Equation (1.2) is optimized with respect to f_0. That is, solve

$$\frac{\partial \ln(\mathcal{L})}{\partial f_0} = 0.$$

The log of Equation (1.2) is

$$\ln \mathcal{L} = -\frac{1}{2}\left(\mathbf{f} - \bar{\mathbf{f}}\right)^T \mathbf{K}^T \left(\mathbf{f} - \bar{\mathbf{f}}\right). - \frac{1}{2}\ln(\det|\mathbf{K}|) - \frac{N}{2}\ln(2\pi) \tag{3.5}$$

By setting the derivative of Equation (3.5) to 0

$$0 = \frac{\partial \mathcal{L}}{\partial f_0} = \frac{\partial \ln(\mathcal{L})}{\partial f_0} = -\mathbf{f}^T\mathbf{K}^{-1}\mathbf{1} - \mathbf{1}^T\mathbf{K}^{-1}\mathbf{f} + 2f_0\left(\mathbf{1}^T\mathbf{K}^{-1}\mathbf{1}\right)$$

is derived. Notice that $\mathbf{K}$ is positive definite symmetric by definition. So, $\mathbf{K}^{-1}$ is also symmetric and

$$\mathbf{1}^T\mathbf{K}^{-1}\mathbf{f} = \mathbf{f}^T\mathbf{K}^{-1}\mathbf{1}.$$

Hence,

$$-\mathbf{f}^T\mathbf{K}^{-1}\mathbf{1} - \mathbf{1}^T\mathbf{K}^{-1}\mathbf{f} + 2f_0\left(\mathbf{1}^T\mathbf{K}^{-1}\mathbf{1}\right) = -2\left(\mathbf{1}^T\mathbf{K}^{-1}\mathbf{f}\right) + 2f_0\left(\mathbf{1}^T\mathbf{K}^{-1}\mathbf{1}\right) = 0.$$

Therefore the maximum likelihood estimate for the prior mean is:

$$f_0 = \frac{\mathbf{1}^T\mathbf{K}^{-1}\mathbf{f}}{\mathbf{1}^T\mathbf{K}^{-1}\mathbf{1}}. \tag{3.6}$$

Note that this analysis relies on using the Gaussian Process likelihood, Equation (1.2), for the prior, and by setting this mean to be a constant vector. Different maximum likelihood estimates can be generated under other assumptions.

Also, this maximum likelihood estimate for the prior mean can be recast as

$$f_0 = \frac{\left(\sum_i \mathbf{K}^{-1}_{[i]}\right) \cdot \mathbf{f}}{\sum_{i,j} \mathbf{K}^{-1}_{[i,j]}}.$$

This interpretation is simply a weighted average with respect to the GP model.

3.2.3 Algorithm

In this upsampling algorithm, single channel grayscale document images are used. The Gaussian Process upsampling algorithm begins with the construction of the model weights with a length scale parameter derived from the original resolution of the image - $\ell = 20 \min(1/h, 1/w)$. The upsampling ratio dictates the number of weight vectors needed, for example, when upsampling $4\times$, 16 new pixels are generated and therefore 16 weight vectors are needed. These vectors are generated by utilizing the Cholesky factorization of $\mathbf{K}$ and then applying back substitution to calculate each $\mathbf{k}_*^T\mathbf{K}^{-1}$. The key factor is that the covariance kernel utilized in this methodology is isotropic– it only depends on the distance between samples. Since a sliding window is used, the upsampling weights only need to be calculated once and can be used throughout the image. This is because the distance between sample pixels are related to their pixel index (i, j). In a similar fashion to the results in Chapter 2, the distance between the pixel that is upsampled and the rest of the window is identical for every window.

When performing upsampling over the document image, a sliding 5×5 pixel window is used as the sample for the GP model. Figure 3.5 helps illustrate the sliding window GP method. The figure contains 3 grids of pixels. The first grid represents the constant maximum likelihood estimate for the prior mean over this pixel grid. The second grid represents the deviation of the sampled pixel

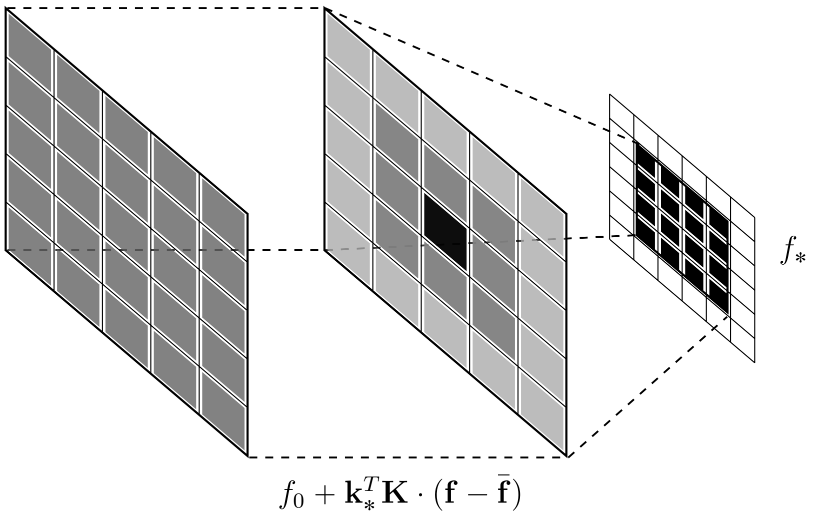

f_*
$f_0 + \mathbf{k}_*^T\mathbf{K} \cdot (\mathbf{f} - \bar{\mathbf{f}})$

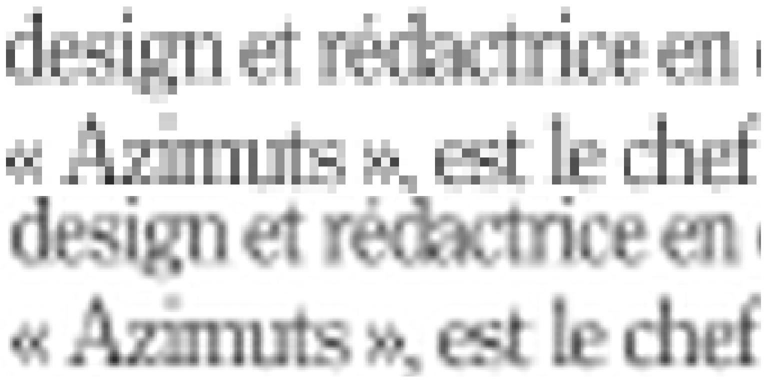
Figure 3.5: Schematic for the 5×5 GP model for 4× upsampling. The completely gray grid illustrates the computation of f_0 over the sample, while the GP model combination on the second grid. The last grid illustrates the 16 new f_* generated by combining the two, effectively replacing the pixel (i, j)

values from the MLE. Together, these grids combine to interpolate 16 new pixels, replacing the pixel in the (i, j) location.

In the implementation of this algorithm, the maximum likelihood estimate for the prior mean is generated when the 5×5 sample is loaded. Then each GP weight vector $\mathbf{k}_*^T\mathbf{K}^{-1}$ is applied to the residual between the MLE and pixels in the sampled window to model the deviation. The deviation and the MLE are combined to generate each new pixel f_*.

As an example, Figure 3.6 is used to illustrate the upsampling results utilizing this Gaussian Processes algorithm. The top image in the figure is the low resolution image (resized by copying the nearest pixels to be the same size as the GP image), and bottom text is from the GP upsampled image. When Tesseract is used on these images, it yields the following texts. The low resolution image Tesseract output is:

"*desigm £rédacimice en fiflanEm, Et le chiet*",


design et rédactrice en

« Azimuts », est le chef

design et rédactrice en

« Azimuts », est le chef

Figure 3.6: Section of Page 154 of the LRDE dataset. Top: 4× downsampled image crop. Bottom: 4× GP upsampled.

which is not an accurate representation of the ground truth. However, for the GP upsampled image, Tesseract generates

"*design et regactnce en « Azzmuts ». est le chef*".

It is clear that the GP upsampled version is much closer to the ground truth text of

"*design et rédactrice en « Azimuts », est le chef*".

Tesseract works best when used on near-binary images as an input. In this case, near binary means that the majority of the pixels in the image are close to 0 if they are within a character, or 255 if not. However, sometimes the single channel images are calculated from RGB images that yield other shades of gray. In this case some images processing techniques can be used to better "binarize" these images. Aside from binarization, images can contain noise or textures within them, which can negatively effect the detection of characters. A common way to handle excess noise and textures is to use a blurring operation to smooth out those regions. However, utilizing these blur convolutions can lead to unwanted removal of edges.

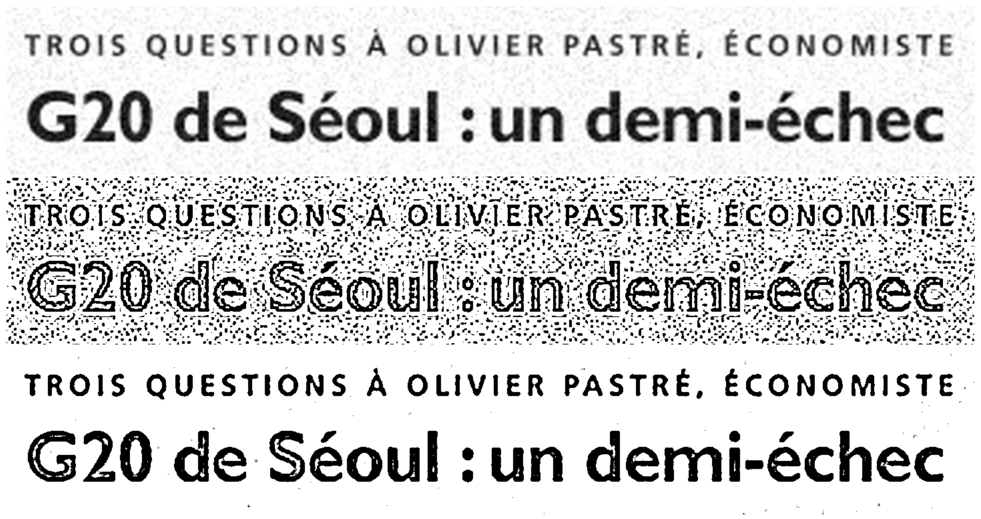
The most utilized blur filter is the Gaussian filter, which is a low-pass filter, dulling out high frequency information, like noise and texture [82]. In the simplest case, a Gaussian blur involves convolving the image with the following discrete kernel:

$$B_g = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{3.7}$$

This Gaussian filter however, has a tendency to smooth edges too much, which can negatively effect the accuracy of the optical character recognition. However, spurious noise and textures can fool an OCR engine into generating false positives – detecting characters that are not present in the image. So instead of Gaussian filtering, the bilateral filtering approach illustrated by Tomasi and Manduchi is used [204]. Bilateral filters reduce noise and textures without compromising edges, that is, without compromising the upsampled edges generated in the GP upsampling.

If the image is not approximately binary, a thresholding technique can be used to force the text to be truly black. An adaptive Gaussian threshold process is used to generate binary images. Thresholding utilizes a set intensity value and replaces all pixels below that value to black and all pixels above the threshold to white. If there are shadows in the image, global thresholding can lead to large portions of the image to be blacked out. This could result in the majority of words in a document image to become inaccessible. An adaptive-thresholding technique utilizes a neighborhood of pixels and calculates the threshold value locally to perform binarization. With Adaptive Gaussian thresholding, the threshold value is the weighted sum of neighborhood pixels in a Gaussian window [96, 27].

Figure 3.7 contains the results of the pipeline for processing low resolution images and is a visual explanation why filtering is necessary, especially when


TROIS QUESTIONS À OLIVIER PASTRÉ, ÉCONOMISTE

G20 de Séoul : un demi-échec

TROIS QUESTIONS À OLIVIER PASTRÉ, ÉCONOMISTE

G20 de Séoul : un demi-échec

TROIS QUESTIONS À OLIVIER PASTRÉ, ÉCONOMISTE

G20 de Séoul : un demi-échec

Figure 3.7: Top: Noisy grayscale GP upsampled text block. Middle: Adaptive thresholding with no filter. Bottom: GP upsampled image with bilateral filter and adaptive thresholding.

performing binarization. The top image is a GP upsampled version of a noisy low resolution image. The middle image is a thresholded version of the noisy image without using the bilateral image filter. Binarization, in this case, enhances the inherent noise, resulting in Tesseract to detect no characters. The bottom image is the noisy input image with bilateral filtering applied, and then thresholded. With the last image the Tesseract engine can detect every character.

The OCR pipeline used is as follows. First, a low resolution image is upsampled using the GP model presented earlier. Then, noise and unwanted textures from the high resolution image are removed while preserving edges by utilizing bilateral filtering. After the GP upsampled image is filtered, if the image is not approximately binary, an adaptive thresholding technique is used to convert the filtered high resolution image into a binary image to be ingested by the Tesseract OCR engine.

For clarity, Figure 3.8 contains an algorithmic diagram with each process.

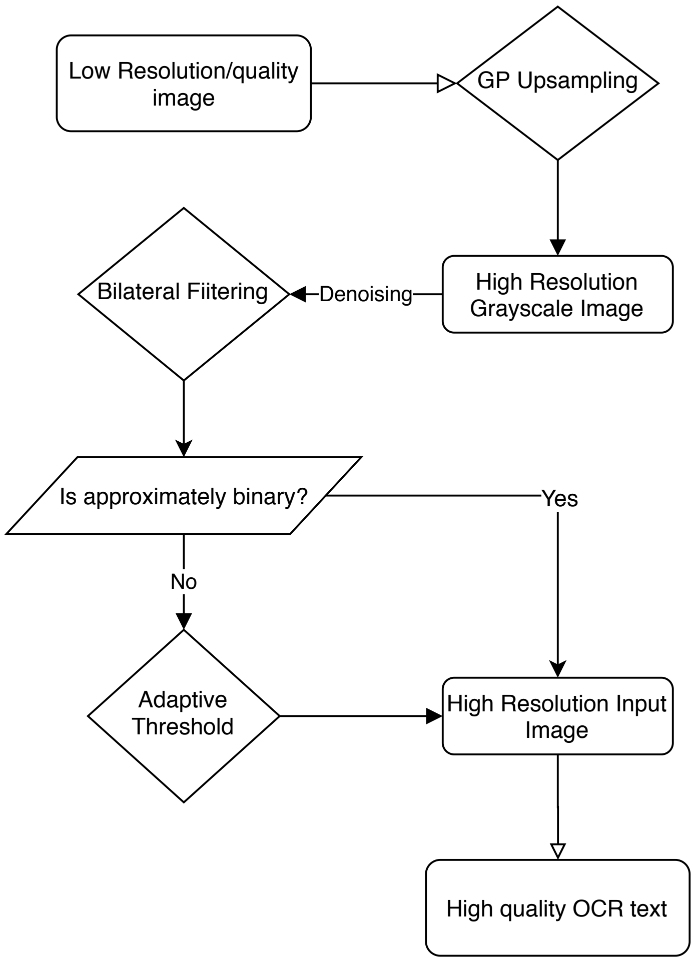


Figure 3.8: The image processing pipeline used for higher quality OCR.

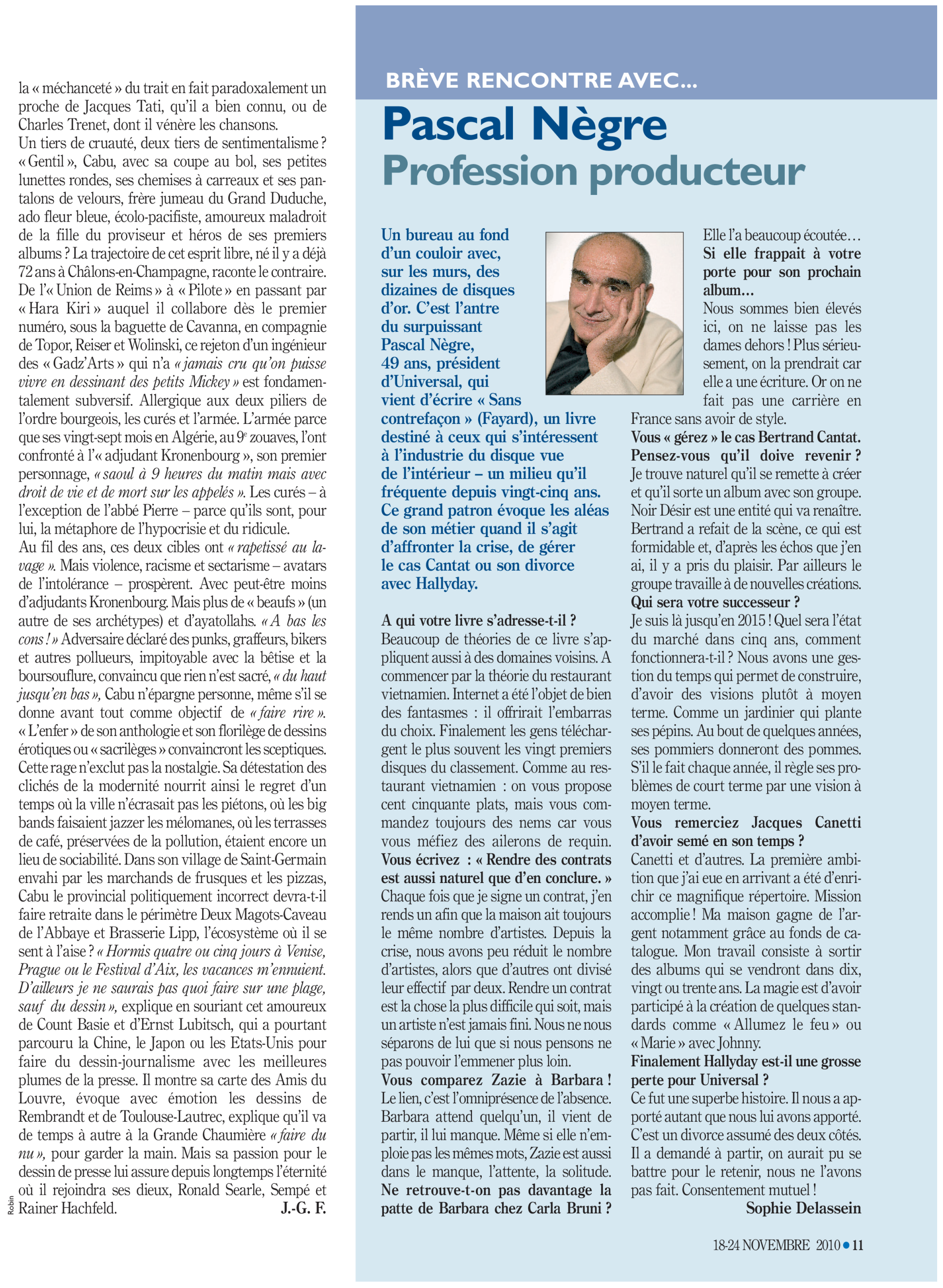
3.3 Testing

In order to test the methodology, the EPITA Research and Development Laboratory (LRDE) dataset from [113] is used. This dataset is publicly available but is copyrighted, ©2012 EPITA Research and Development Laboratory (LRDE) with permission from Le Nouvel Observateur. This dataset is based on the French magazine Le Nouvel Observateur, issue 2402, November 18th-2th, 2010. The original images come from this magazine, and LRDE has generated the ground truth OCR from these images. This dataset is free for research, evaluation, and illustration and can be downloaded from LRDE's website: https://www.lrde.epita.fr/dload/olena/datasets/dbd/1.0/. For illustration purposed, Figure 3.9 contains a page of the magazine that is used for testing.

To test the proposed GP upsampling algorithm, the original images' resolution is downsampled 4× in width and height. Then these low resolution representations are combined with Gaussian noise. Next, the noisy low resolution images are upsampled using the GP method illustrated in this chapter. Finally, the upsampled images are then passed through the image processing pipeline illustrated in Figure 3.8, to extract detected characters.

For this purpose, accuracy is calculated by comparing the number of words detected in the upsampled document to those that are present in the ground truth text. This is a fairly conservative measure, as increased accuracy in upsampling can lead to increased similarity in generated words with the true words. However, in this case, number of true words matched is a more direct measurement of accuracy that will effect applications that utilize image extracted text.

First, the accuracy of the GP method is compared to the OCR extracted utilizing the low resolution images. Figure 3.10 contains a graph comparing the accuracy of OCR obtained from the GP upsampled images against OCR from the



la « méchanceté » du trait en fait paradoxalement un proche de Jacques Tati, qu'il a bien connu, ou de Charles Trenet, dont il vénère les chansons.
Un tiers de cruauté, deux tiers de sentimentalisme ? « Gentil », Cabu, avec sa coupe au bol, ses petites lunettes rondes, ses chemises à carreaux et ses pantalons de velours, frère jumeau du Grand Duduche, ado fleur bleue, écolo-pacifiste, amoureux maladroit de la fille du proviseur et héros de ses premiers albums ? La trajectoire de cet esprit libre, né il y a déjà 72 ans à Châlons-en-Champagne, raconte le contraire. De l'« Union de Reims » à « Pilote » en passant par « Hara Kiri » auquel il collabore dès le premier numéro, sous la baguette de Cavanna, en compagnie de Topor, Reiser et Wolinski, ce rejeton d'un ingénieur des « Gadz'Arts » qui n'a *« jamais cru qu'on puisse vivre en dessinant des petits Mickey »* est fondamentalement subversif. Allergique aux deux piliers de l'ordre bourgeois, les curés et l'armée. L'armée parce que ses vingt-sept mois en Algérie, au 9e zouaves, l'ont confronté à l'« adjudant Kronenbourg », son premier personnage, *« saoul à 9 heures du matin mais avec droit de vie et de mort sur les appelés »*. Les curés – à l'exception de l'abbé Pierre – parce qu'ils sont, pour lui, la métaphore de l'hypocrisie et du ridicule.
Au fil des ans, ces deux cibles ont *« rapetissé au lavage »*. Mais violence, racisme et sectarisme – avatars de l'intolérance – prospèrent. Avec peut-être moins d'adjudants Kronenbourg. Mais plus de « beaufs » (un autre de ses archétypes) et d'ayatollahs. *« A bas les cons ! »* Adversaire déclaré des punks, graffeurs, bikers et autres pollueurs, impitoyable avec la bêtise et la boursouflure, convaincu que rien n'est sacré, *« du haut jusqu'en bas »,* Cabu n'épargne personne, même s'il se donne avant tout comme objectif de *« faire rire »*. « L'enfer » de son anthologie et son florilège de dessins érotiques ou « sacrilèges » convaincront les sceptiques. Cette rage n'exclut pas la nostalgie. Sa détestation des clichés de la modernité nourrit ainsi le regret d'un temps où la ville n'écrasait pas les piétons, où les big bands faisaient jazzer les mélomanes, où les terrasses de café, préservées de la pollution, étaient encore un lieu de sociabilité. Dans son village de Saint-Germain envahi par les marchands de frusques et les pizzas, Cabu le provincial politiquement incorrect devra-t-il faire retraite dans le périmètre Deux Magots-Caveau de l'Abbaye et Brasserie Lipp, l'écosystème où il se sent à l'aise ? *« Hormis quatre ou cinq jours à Venise, Prague ou le Festival d'Aix, les vacances m'ennuient. D'ailleurs je ne saurais pas quoi faire sur une plage, sauf du dessin »,* explique en souriant cet amoureux de Count Basie et d'Ernst Lubitsch, qui a pourtant parcouru la Chine, le Japon ou les Etats-Unis pour faire du dessin-journalisme avec les meilleures plumes de la presse. Il montre sa carte des Amis du Louvre, évoque avec émotion les dessins de Rembrandt et de Toulouse-Lautrec, explique qu'il va de temps à autre à la Grande Chaumière *« faire du nu »,* pour garder la main. Mais sa passion pour le dessin de presse lui assure depuis longtemps l'éternité où il rejoindra ses dieux, Ronald Searle, Sempé et Rainer Hachfeld.

J.-G. F.

Robin

BRÈVE RENCONTRE AVEC…

Pascal Nègre
Profession producteur

Un bureau au fond d'un couloir avec, sur les murs, des dizaines de disques d'or. C'est l'antre du surpuissant Pascal Nègre, 49 ans, président d'Universal, qui vient d'écrire « Sans contrefaçon » (Fayard), un livre destiné à ceux qui s'intéressent à l'industrie du disque vue de l'intérieur – un milieu qu'il fréquente depuis vingt-cinq ans. Ce grand patron évoque les aléas de son métier quand il s'agit d'affronter la crise, de gérer le cas Cantat ou son divorce avec Hallyday.

A qui votre livre s'adresse-t-il ?
Beaucoup de théories de ce livre s'appliquent aussi à des domaines voisins. A commencer par la théorie du restaurant vietnamien. Internet a été l'objet de bien des fantasmes : il offrirait l'embarras du choix. Finalement les gens téléchargent le plus souvent les vingt premiers disques du classement. Comme au restaurant vietnamien : on vous propose cent cinquante plats, mais vous commandez toujours des nems car vous vous méfiez des ailerons de requin.
Vous écrivez : « Rendre des contrats est aussi naturel que d'en conclure. »
Chaque fois que je signe un contrat, j'en rends un afin que la maison ait toujours le même nombre d'artistes. Depuis la crise, nous avons peu réduit le nombre d'artistes, alors que d'autres ont divisé leur effectif par deux. Rendre un contrat est la chose la plus difficile qui soit, mais un artiste n'est jamais fini. Nous ne nous séparons de lui que si nous pensons ne pas pouvoir l'emmener plus loin.
Vous comparez Zazie à Barbara !
Le lien, c'est l'omniprésence de l'absence. Barbara attend quelqu'un, il vient de partir, il lui manque. Même si elle n'emploie pas les mêmes mots, Zazie est aussi dans le manque, l'attente, la solitude.
Ne retrouve-t-on pas davantage la patte de Barbara chez Carla Bruni ?
Elle l'a beaucoup écoutée…
Si elle frappait à votre porte pour son prochain album…
Nous sommes bien élevés ici, on ne laisse pas les dames dehors ! Plus sérieusement, on la prendrait car elle a une écriture. Or on ne fait pas une carrière en France sans avoir de style.
Vous « gérez » le cas Bertrand Cantat. Pensez-vous qu'il doive revenir ?
Je trouve naturel qu'il se remette à créer et qu'il sorte un album avec son groupe. Noir Désir est une entité qui va renaître. Bertrand a refait de la scène, ce qui est formidable et, d'après les échos que j'en ai, il y a pris du plaisir. Par ailleurs le groupe travaille à de nouvelles créations.
Qui sera votre successeur ?
Je suis là jusqu'en 2015 ! Quel sera l'état du marché dans cinq ans, comment fonctionnera-t-il ? Nous avons une gestion du temps qui permet de construire, d'avoir des visions plutôt à moyen terme. Comme un jardinier qui plante ses pépins. Au bout de quelques années, ses pommiers donneront des pommes. S'il le fait chaque année, il règle ses problèmes de court terme par une vision à moyen terme.
Vous remerciez Jacques Canetti d'avoir semé en son temps ?
Canetti et d'autres. La première ambition que j'ai eue en arrivant a été d'enrichir ce magnifique répertoire. Mission accomplie ! Ma maison gagne de l'argent notamment grâce au fonds de catalogue. Mon travail consiste à sortir des albums qui se vendront dans dix, vingt ou trente ans. La magie est d'avoir participé à la création de quelques standards comme « Allumez le feu » ou « Marie » avec Johnny.
Finalement Hallyday est-il une grosse perte pour Universal ?
Ce fut une superbe histoire. Il nous a apporté autant que nous lui avons apporté. C'est un divorce assumé des deux côtés. Il a demandé à partir, on aurait pu se battre pour le retenir, nous ne l'avons pas fait. Consentement mutuel !

Sophie Delassein

18-24 NOVEMBRE 2010 • **11**

Figure 3.9: Page 11, *Le Nouvel Observateur*, before grayscale conversion.

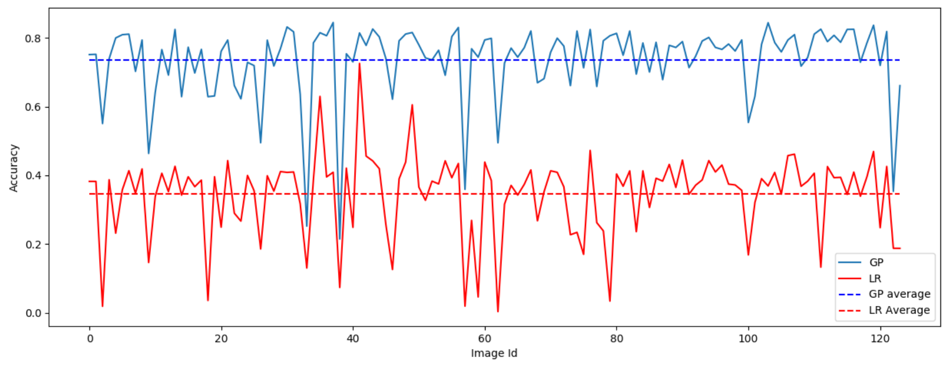


Figure 3.10: GP upsampled OCR accuracy vs the Low Resolution accuracy with dashed lines denoting average accuracies.

low resolution images, for each image in the dataset. In the figure, the blue line represents the OCR accuracy for each GP upsampled image, whereas the red line is the OCR accuracy of the low resolution images. Flat dashed lines are included to illustrate the mean accuracy of each set. There are several dips in the graph where both the upsampled accuracy and the low resolution accuracy are very low, these pages of the magazine are comprised of mostly images where text is not the dominant feature. The extraneous information limits the capabilities of the Tesseract OCR engine.

Most applications that require OCR will upsample sufficiently low resolution images. So, naturally, the GP algorithm is compared against the bicubic interpolation method, a common baseline in upsampling algorithms. For this implementation the bicubic method used is contained in the Python Image Library [106]. In this test, the text generated by the GP based pipeline is compared against an analogous bicubic interpolation based pipeline. Figure 3.11 contains a plot of the relative gain in accuracy when utilizing GP over bicubic interpolation over the LRDE dataset. In the figure, the relative gain is depicted by the blue dots for each image in the dataset. Additionally, a line denoting equal performance is

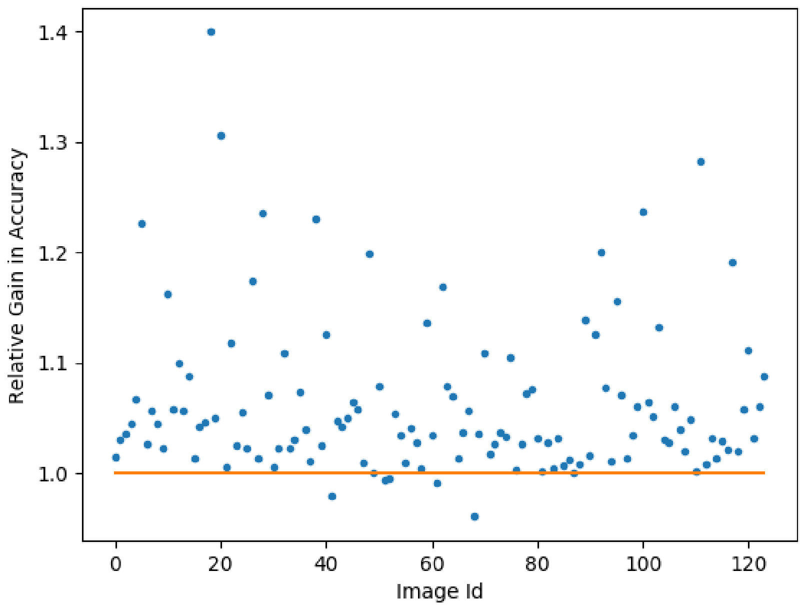


Figure 3.11: The relative gain utilizing GP upsampling vs bicubic over the noisy low resolution test set. The blue dots are the individual accuracy gains, and a reference line corresponding to equal accuracy is plotted in orange.

plotted as an orange line for reference. For the majority of images, the proposed algorithm's extracted text better matches the ground truth text over the baseline interpolation.

Some summary statistics are included in Table 3.1. The GP algorithm performs the best over the base low resoution images, and the bicubic interpolation based pipeline. The GP algorithm had the highest average accuracy, lowest variance and the highest minimum and maximum accuracy out of the three tests. The last column in the table is the relative gain in OCR accuracy by using the GP algorithm instead of Bicubic or just using the low resolution image. There is a 6.26% increase in character recognition against the bicubic upsampling.

Table 3.1: Summary statistics of the OCR accuracy over the LRDE subsampled dataset.

	Average	Variance	Max	Min	GP Relative Increase
GP	0.735020	0.012018	0.844515	0.214765	N/A
Bicubic	0.695874	0.013746	0.835996	0.175597	6.26%
Low Resolution	0.345170	0.014018	0.725663	0.003584	195%

3.4 Chapter Summary

In this chapter, a new Gaussian Process based interpolation model was produced for the explicit purpose of upsampling single-channel document images. Testing over a real-word data set revealed an increase in OCR accuracy over the baseline upsampling method, bicubic interpolation, when used in conjunction with the Tesseract OCR engine.

One could build a Gaussian Process model over the entire low resolution image and generate new pixels with inputs in a non-local sense. This provides issues in multiple areas. The kernel utilized in this context decay rapidly as distance is increased, so the new information gained will become less of a contribution than a hinderance when it comes to computation. Even though the weights are calculated using the Cholesky Factorization of the covariance matrix $\mathbf{K}$, the computational complexity of factorization is still $n^3/3$ where n is the size of row and column size of $\mathbf{K}$ [210]. So even on a relatively small resolution image, say 500×500, $\mathbf{K}$ will have size 250000, which will require 5.208×10^{15} operations. This is realistically infeasible, which leads well into the approach described in this chapter. The windowed GP model can be reinterpreted as a Sparse Gaussian Processes that only utilizes information that is local to the interpolation pixels, which will have the most relevant information in both models.

Some minor improvements could be gained by optimizing the length scale parameter, which could be found by maximizing the log-likelihood with respect to ℓ. However, each window may have a different optimal length scale, which again, leads to an unwanted increase in computational complexity. Additionally, one can tune ℓ for the dataset, but the value in this chapter appears to be general, as it depends on the size of the low resolution image.

Utilizing the proposed GP algorithm as an upsampling method for Optical Character Recognition yields on average a positive gain in accuracy versus a more traditional bicubic method when used to upsample the images for inputs to the Tesseract OCR engine. The GP algorithm uses a sliding window of 5×5 pixel sampled across the image. The yield in accuracy against bicubic can help text based Natural Language Processing (NLP) models become perform better when placed in an end-to-end environment, like in financial applications, or for accessibility of documents and scanned images for people who are visually impaired.

Chapter 4

A Composite Gaussian Process-Convolutional Neural Network Model for Single Image Super Resolution

The previous chapter presented a method based solely on Gaussian Processes to upsample document images for the express purposes of enhancing word accuracy from Optical Character Recognition. For this purpose, gaining a model for pixels representing letters is used. However, Single Image Super Resolution (SISR) is a far more generalized process.

Super resolution models wish to derive images that regain perceptual validity and textures rather than enabling an OCR engine to detect characters. In this case, general images can be far more complex than document images, and do not have a-priori known tendencies or features. Keeping this in mind, a new hybrid Gaussian Process Convolutional Neural Network approach is employed.

This chapter begins with a brief on SISR as well as the some of the common methods to tackle this difficult problem. Following this introduction, the proposed Gaussian Process upsampling model is introduced, along with the dataset that will be used for training and testing. Also the evaluation metrics and a comparison between the proposed GP based model and the baseline bicubic method is discussed in this section. Following this discussion, the Additive Enhancement Network is introduced with details on the architecture and training. Next, the proposed Gaussian Process upsampling model with Additive Enhancement Network (GPAEN) is evaluated against some of the other methods in the state-of-the-art. Lastly, this chapter culminates on a discussion of the efficacy of GPAEN against the other models and what it can be used for in conjunction with SISR.

4.1 Introduction

Single Image Super Resolution is a complex problem that has challenged the field of computer vision and image processing. This problem is the attempt at generating a high resolution-high fidelity image from a low resolution image. Images, in general, are digital representations of reality and can be thought of as coming from complicated function spaces. Sighted humans are visual creatures, so naturally attempts to recreate high fidelity representations from low resolution images have tantalized us since the use of digital images.

Due to the possibly complex details in high resolution images, standard interpolation methods (polynomial based methods for example), no longer encompass the state-of-the-art. Most interpolation methods are based on utilizing aspects of the underlying function space, often $\mathcal{C}^{\infty}$, like the solutions to differential equations. Conversely, images do not necessarily belong to $\mathcal{C}^{\infty}$, and often are not continuous. Instead, strong discontinuities encompassing (e.g. high contrast and textures) can be found in interesting images. Aspects that were garnered in the previous chapter have inspired some of the methodology in this chapter.

Gaussian Processes have been used for SISR before, but in the context of GP regression to correct low order upsampling of images [86, 83]. In these papers, a regular polynomial based interpolation method is used for upsampling, and then GP regression is used to enhance the data to be more realistic. In the last decade however, example based artificial intelligence methods such as Convolutional Neural Networks (CNNs) have dominated the leading edge in the single image super resolution problem [53, 52, 95, 165, 108, 126]. These approaches learn the SISR operation by iterating though low resolution images as inputs are comparing the result of the NN to the ground truth high resolution image.

More recently, complex models involving Generative Adversarial Networks (GANS) have been employed to great success exemplified by the seminal paper [115]. GANs utilize two neural networks (NN), one NN to generate objects, and another to discriminate real objects and the objects produced by the generator. The system of NNs are trained when the discriminator cannot determine whether the generator's objects are members of the ground truth. In the case of super resolution, the generator NN attempts to generate high resolution representations of the images and the discriminator asserts if the generator's image is a member of the ground truth dataset. When the discriminator can no longer tell the difference between ground truth images and the generator's, the generator is used as a NN to perform super resolution.

4.2 Method

Single image Super Resolution can be thought as a combination of two operations, upsampling and enhancement. The majority of NN based SISR algorithms attempt to combine these two operations into one. This has some advantages, the network will learn an upsampling method that works best on the data, and can enhance as needed. However, in this case, the network can be more tied to the loss function, and more standard and well understood loss functions can lead to unwanted results. Because of this, some researchers have moved to create new and exotic loss functions, such as the texture loss [76] to be used with GANs.

The method presented here does not attempt to combine the two operations into a single network. Instead a GP interpolation method is used to upsample the image data, and then a CNN is used to enhance the upsampled image. The GP upsampling model shares the covariance kernel with the GP method in Chapter 3. However, in this context, there is no real apparent pattern within the data, so a zero mean function will be used instead of an analogous maximum likelihood estimate.

Rather than input theGP upsampled image into CNN and retrieve the an enhanced version as an output, the CNN is used to predict the error between the image generated by GP upsampling and the true high resolution image. Essentially, the aim is to produce following formula:

$$\mathcal{I}_{gt} = \mathcal{I}_{GP} + \mathcal{E}(\mathcal{I}_{GP}) \tag{4.1}$$

where the error term $\mathcal{E}(\mathcal{I}_{GP})$ is approximated using

$$\mathcal{E}(\mathcal{I}_{GP}) \sim CNN(\mathcal{I}_{GP}). \tag{4.2}$$

With this approach training is done on the error instead of the image. This can further help reduce the tendency for the model to be stuck into a local minimum, as the upsampled image is, at first glance, a good approximation to the high resolution image.

The Diverse 2K data from the NTIRE 2017-2018 challenges [5, 198] is used for training and testing the proposed method. This dataset is comprised of 900 images with over 2K resolution for the ground truth, and two downsampled versions, one with a 2× reduction in resolution (in width and height) and the other with a 4× reduction in resolution. Note that in this chapter as with the previous chapter, 2× is in both width and height, so 4 pixels are generated in the place of one, likewise with the 4× upsampling 16 pixels are generated.

4.2.1 Gaussian Process Upsampling Model

As previously mentioned, a Gaussian Process model is employed to upsample a low resolution input image. In this model, a sample 3×3 pixel sample is used and is windowed across the input image. The posterior mean is used to generate newly upsampled pixels as in Equation (1.4) using the sample data from this window. In this chapter, a zero mean Gaussian Process with the Matèrn covariance function with $\nu = 3/2$, as illustrated in Equation (3.3), is used – in contrast to the models utilized in the previous chapters. The size of the sample window was reduced for better speed, and because an enhancing convolutional neural network will be used.

An example schematic for the pixels in and out of the GP model can be found in Figure 4.1. The low resolution sample block is represented by the gray cells with the black cell in the center in the diagram. The 4× newly upsampled pixels are shown in black replacing the center pixel from the low resolution sample. Like in the previous chapters, the model weights are constructed before the processing of the image. Additionally, every new pixel is updated following the formula

$$\hat{I}(\tilde{i},\tilde{j}) = \mathbf{w}_{\tilde{i},\tilde{j}}^{T} I_S \tag{4.3}$$

here $\mathbf{w}_{i\tilde{i},\tilde{j}}$ is the GP model weight vector for the new pixel at index $(\tilde{i},\tilde{j})$. In general there are $r_w r_h$ new pixels to be predicted, where r_w, r_h are the upsampling factors for the height and width of the image respectively. In this formulation $\hat{I}$ is the GP uspampled image, and I_S is the 3×3 sample pixel patch with indices ranging from $i-1$ to $i+1$ and $j-1$ to $j+1$ for height and width respectively. The GP model weight vector is generated using

$$\mathbf{w}_{\tilde{i},\tilde{j}} = \mathbf{k}_{\tilde{i},\tilde{j}}^{T} \mathbf{K}^{-1} \tag{4.4}$$

in a similar fashion to the previous chapters. The covariance vector is produced by calculating the covariance at the point of interest and along the sample:

$$k_{\tilde{\mathbf{i}}}(\mathbf{i}) = K_{3/2}(\tilde{\mathbf{i}},\mathbf{i})$$

where $\mathbf{i} = (i,j)$ and $\tilde{\mathbf{i}} = (\tilde{i},\tilde{j})$. This further illustrates that each weight is spatially constructed, so one can cast the interpolation as an image convolution by reshaping the weight vectors into a convolution kernel. With this interpretation, $r_h r_w$ convolutions are applied to the low resolution image in order to generate the upsampled image. This can be stipulated in a per-pixel format,

$$\hat{I}(\tilde{i},\tilde{j}) = \mathbf{W}_{\tilde{i},\tilde{j}} \circledast I_S \tag{4.5}$$

where $\mathbf{W}_{\tilde{i},\tilde{j}}$ is the convolution kernel to produce the RGB pixels at index $\tilde{i},\tilde{j}$. Then the upsampled image is just the amalgamation of these convolved images.

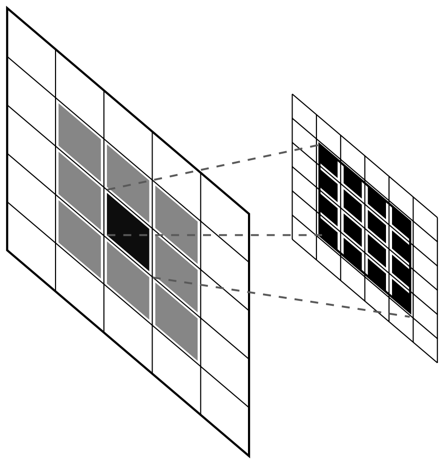


Figure 4.1: An illustrative schematic for the 3×3 GP model for 4× image upsampling.

To illustrate the use of this GP based upsampling operation, Figure 4.2 contains a low resolution crop from image 0028 in the Div2K dataset and the 4× GP upsampled version. The 4× GP model upsamples the image well, preserving edges and enhancing detail. However, the super resolution is not complete, there are many details that are not apparent in the upsampled image that would be present in a natively high resolution image. Detailed textures that are not exhibited in the low resolution image will not be able to be generated by GP alone. To generate a more high fidelity image an operation that distills the fundamental information that is missed under the GP model is required.

Comparison with Bicubic

To assess the quality of the proposed method, a comparison to the commonly-used baseline interpolation method, bicubic, is made. A metric called the peak signal to noise ratio (PSNR) is commonly used to compare the quality of images. This metric, however, is not the only metric to describe the quality of a representation of an image. In this chapter, sharpness is also a metric that is used to test

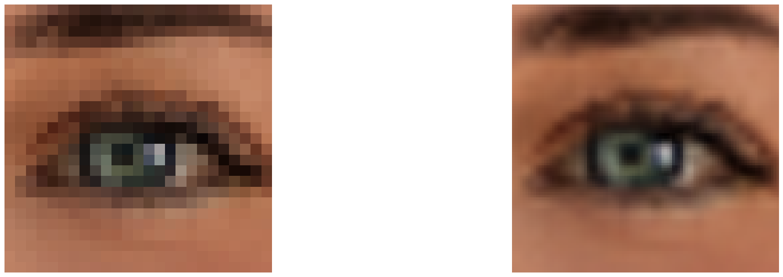

(a) **(b)**

Figure 4.2: a): Selected crop from image 0028 of the Div2k dataset downsampled 4×. b): The LR crop upsampled by the GP method to its original resolution.

the SISR methods.

In formula, PSNR (measured in dB) is

$$PSNR = 10 \cdot \log_{10}\left(\frac{\Omega^2}{MSE}\right) = 20 \cdot \log_{10}(\Omega) - 10 \cdot \log_{10}(MSE), \tag{4.6}$$

here Ω is the max possible value of the image type. MSE is the mean squared error

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left(I(i,j) - \hat{I}(i,j)\right)^2 \tag{4.7}$$

where I is the ground truth, and $\hat{I}$ is the approximation, and m, n are the width and height of the image.

Sharpness in this dissertation is calculated as the variance of the Laplacian operator applied to the grayscale version of an image. That is,

$$Sharp = \mathbb{E}\left[\left(\nabla^2 G\right)^2\right] - \mathbb{E}^2\left[\nabla^2 G\right] \tag{4.8}$$

where G is the grayscale transformation of I, ∇^2 is the Laplacian operator for images, and $\mathbb{E}$ is the 'expectation' operator. The standard 3×3 Laplacian filter kernel is

$$\mathbf{L} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{4.9}$$

when convolved with an image the result is analogous to the numerical Laplacian. That is,

$$\nabla^2(G)(i,j) = \sum_{k=0}^{2}\sum_{l=0}^{2} L(k,l)G(i-(k-1), j-(l-1)). \tag{4.10}$$

The grayscale transformation is the weighted sum that is computed over the each channel for each pixel, that is,

$$G(i,j) = 0.299I_r(i,j) + 0.587I_g(i,j) + 0.114I_b(i,j) \tag{4.11}$$

which follows the CCIR 601 luma format. The subscripts represent the color channel for the image I, I_r is red, I_g is green, etc. To put sharpness into context the sharpness of the each image is divided by the sharpness of the ground truth image, to yield a comparative sharpness metric. In this metric, closer to 1 is more desirable. The downsampling operation greatly reduces the sharpness of the image, effectively applying a low-pass filter to the image. The image upsampling method that better constructs the sharpness of the original image will have a higher comparative sharpness value. Comparative Sharpness is another good measure of upsampling performance, since downsampling typically acts as a low-pass filter and removes high frequency information. Effectively, comparative sharpness examines the intensity of the edges and textures in an image as compared to the original image. Other definitions of sharpness have been proposed, most notably utilizing Sobel gradients in x and y [67].

To illustrate the use of sharpness, Figure 4.3 contains 3 Laplacian filtered

(a) HR (CompSharp) **(b)** GP 2× (0.1463) **(c)** Bicubic 2× (0.1174)

Figure 4.3: Grayscale Laplacian Crop section of Image 0824 in the DIV2K validation set (color inverted for better visualization).

Table 4.1: Average Peak Signal to Noise Ratio and Comparative Sharpness of the bicubic and GP methods over the training portion of the DIV2K dataset.

	Upsample 2x	**Upsample 4x**
GP PSNR	31.234	26.649
Bicubic PSNR	31.171	26.653
GP Comparative Sharpness	0.1956	0.0425
Bicubic Comparative Sharpness	0.1636	0.0243

grayscale images. These images are a section of image 0824 in the DIV2K dataset. The first image is of the filtered grayscale of the true high resolution image. The second image (center in the figure) is the GP 2× upsampled version. The final image is the Laplacian filter applied to the bicubic 2× upsampled representation. In this figure, the filtered GP image has greater intensity than the bicubic upsampled image.

When compared to bicubic interpolation, it is found that for the equivalent peak signal to noise ratio (PSNR), the sparse GP interpolation yields higher comparative sharpness values. Table 4.1 contains some interesting statistics when used over the 2K dataset, namely the average PSNR and comparative sharpness values.

For a more in-depth look into the PSNR and sharpness for each interpolation, see Figure 4.4 for the 2× values and Figure 4.5 for the 4× metrics. In these figures, PSNR and comparative sharpness is calculated over the training portion of the 2K dataset, the blue and orange dots represent the values for the GP method and bicubic respectively. In both upsampling cases, the PSNR for this GP model and the bicubic interpolation are of the same order, but the sharpness of the GP upsampled images are much greater. In the 4× upsampling case, the bicubic method barely surpasses the GP upsampling method in PSNR, but the GP method has much greater comparative sharpness. However, in the 2× upsampling case, the GP method surpasses bicubic in PSNR and comparative sharpness as presented in Table 4.1. Furthermore, in Figures 4.4 and 4.5, the comparative sharpness is much higher than the bicubic interpolation method across the dataset.

4.2.2 Additive Enhancement Convolutional Neural Network

In order to further enhance the super-resolved images upsampled with the GP method, a Convolution Neural Network is built. As discussed briefly before, this CNN will utilize patches of the GP-upsampled image and generate a ’guess’ at the difference between the modeled image and the ground truth. A form of residual CNN’s have been used to great effect in image classification with the seminal creation of ResNet [87], and has been used for SISR as well [126], in a somewhat different context. In these networks, layer blocks are combined with their inputs and this is what gives their name.

In this algorithm, the CNN will approximate the residual between correspond-

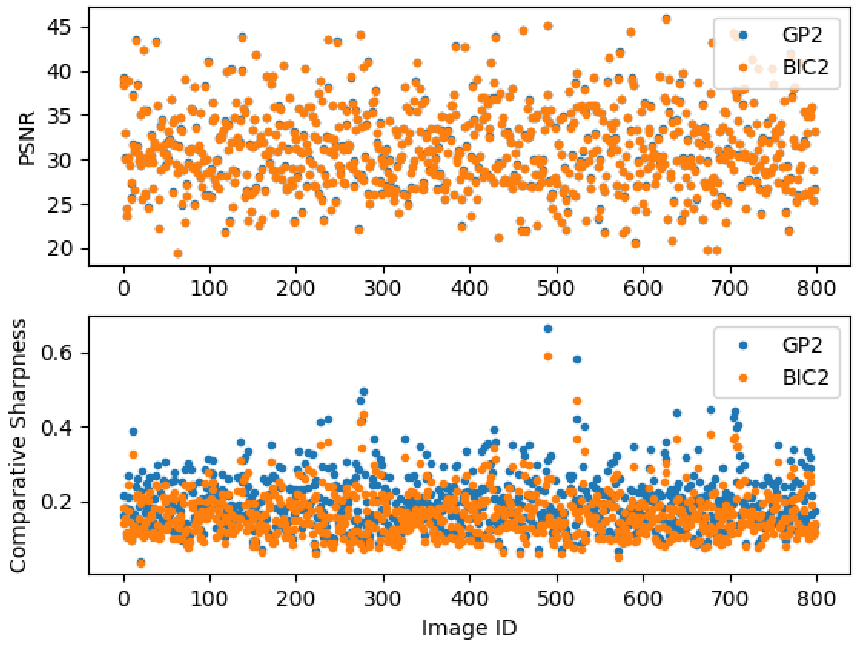


Figure 4.4: Peak Signal To Noise Ratio and Comparative Sharpness Values for Bicubic and GP for 2× upsampling.

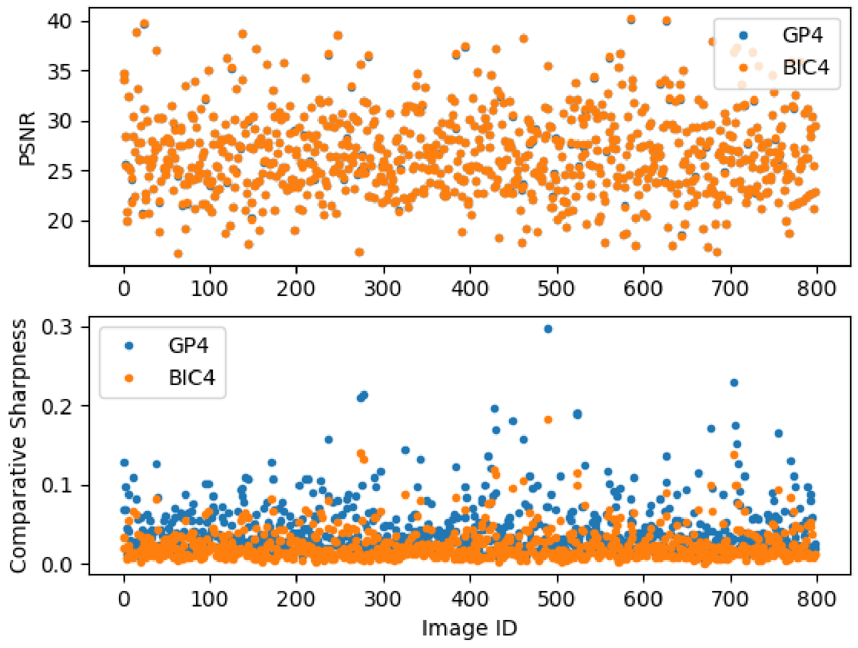


Figure 4.5: Peak Signal To Noise Ratio and Comparative Sharpness Values for Bicubic and GP for 4× upsampling.

ing patches of the GP upsampled image, $\hat{I}$ and the ground truth, I_{HR}.

$$F(\hat{I}) \approx I_{HR} - \hat{I} \tag{4.12}$$

This plan is illustrated pictorially in Figure 4.6 with a crop of image 0235 of the Div2K data set. The top left picture is the low resolution crop, and the GP upsampled version is directly right of it. On the bottom of the figure, a normalized version of the residual between the ground truth (pictured bottom right) and the GP upsampled version is displayed. The ground truth contains high frequency information (mostly textures) that the GP model cannot infer well from the low resolution examples and this is exemplified in the residual. This is where the CNN will come in, to create a series of image convolutions that will accentuate the GP upsampled image into an image that more closely resembles the ground truth. The network will learn to predict textures and further enhance edges. It is stressed that the residual presented in Figure 4.6 is scaled for better visualization. The intensities of residual are actually much lower and centered about 0.

Brief on Convolutional Neural Networks

Before the Additive Enhancement Network in described in detail, a brief summary of convolutions neural networks is provided. Convolutional Neural Networks are artificial neural networks whose layers are convolutions [171]. A convolutional layer, in the simplest case, with input size (N, C_x, H_x, W_x) and output size (N, C_y, H_y, W_y) can be precisely described and formulated as

$$y(N_i, C_j) = \beta(C_j) + \sum_{k=0}^{C_x-1} \Big[w(C_j, k) * x(N_i, k) \Big]. \tag{4.13}$$

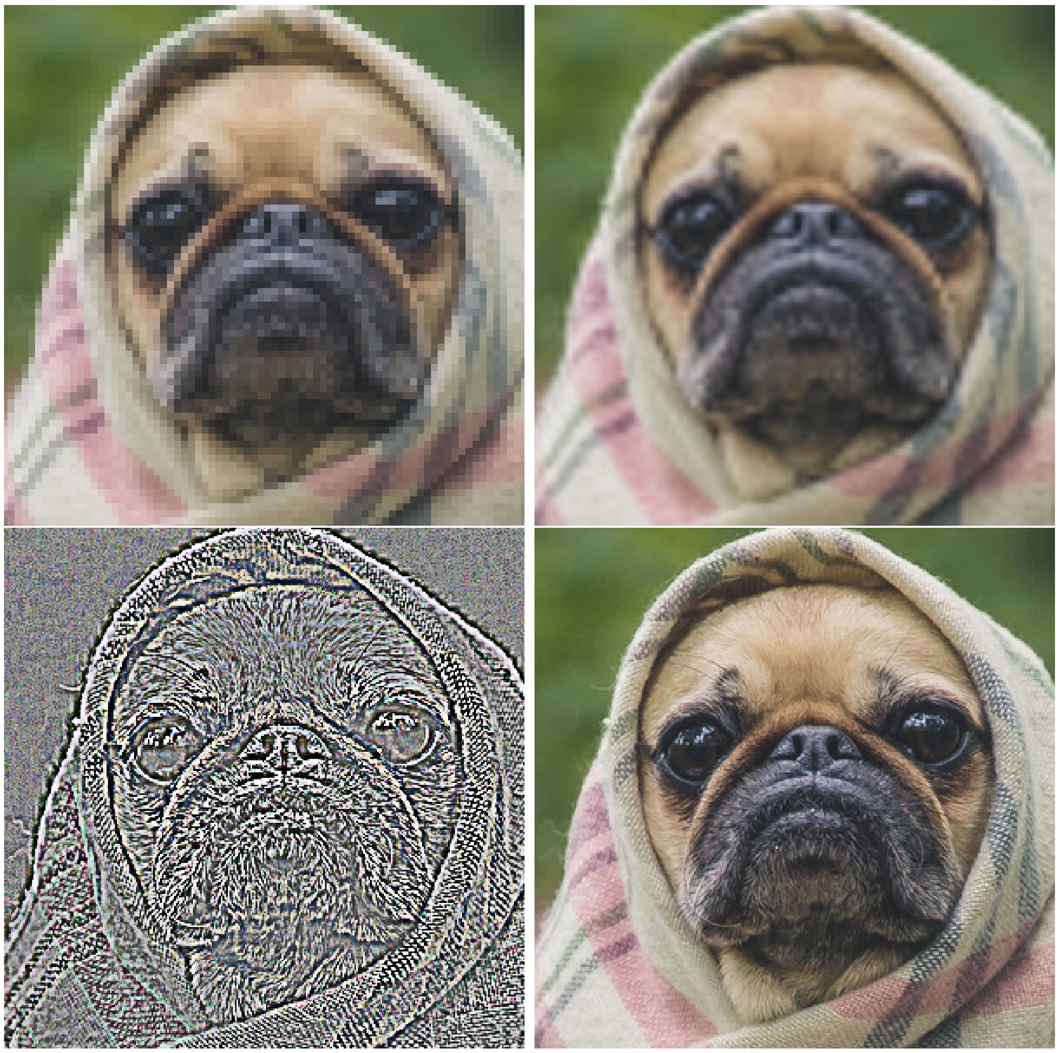


Figure 4.6: Crop of image 0235 in the Div2K dataset. Top left: Low Resolution. Top right: GP 4x Upsampled. Bottom left: Normalized Residual between GP and ground truth. Bottom right: Ground truth image.

In this equation, x is the input to the convolutional layer, and y is the output. The variable x has a size of (N, C_x, H_x, W_x), where N is the batch size, C_x is the number of channels that x has and H_x, W_x are the height and width of x respectively. Analogously, y has size (N, C_y, H_y, W_y). The formula in Equation 4.13, is for the i^{th} object in the batch and the j^{th} channel in the output channels. Also, $*$ is the discrete convolution operator with respect to the convolution kernel $w(C_j, k)$, and $\beta(C_j)$ is the bias for channel j. Essentially, a convolutional layer is comprised of linear combinations of the input data and convolutional weights and a bias term. Each $\beta(C_j)$ and $w(C_j, k)$ are the trainable parameters discovered when optimized over the dataset.

The convolutional layer is generally paired with some form of nonlinearity, called an activation function [146]. For regression type problems, like Single Image Super Resolution, convolutional layers are generally pared with activation functions like the rectified linear unit(ReLU), which is a linear function if the weight is positive and 0 else. Variants on this function exist like the leaky ReLU, and the exponential linear unit. These all feature diminished range for negative weights [146]. Furthermore, these activation functions seek to enhance features that are exposed by the convolutions.

In image based applications of deep learning, convolutional layers are often stacked to create deep neural networks and infer features from the dataset or an operation. The number of layers to couple and the general architecture of the CNN are hyper-parameters that one must consider on an application basis.

Neural Network Architecture

Arguably the most important part of a building a neural network is deciding on the architecture. For this application of enhancing the GP upsampled image a deep convolutional neural network is utilized.

Using the entire GP upsampled image as an input for the enhancement network is impractical and not generalizable. Instead a 0 padded version of the GP image is reshaped into an array of 128×128 patches, each patch processed by the CNN. The 128×128 windows are passed into the CNN comprised of an input convolutional layer, followed by 6 residual blocks, and a channel reduction convolutional output layer. Each residual block is formulated as follows. The tensor is fed into a convolutional layer which is then followed by exponential linear unit [146]. The output of these operations is processed by a second convolutional layer, ending the block. The output of each block is summed with the output of the previous block. To better illustrate this concept, Figure 4.7 contains a diagram of the residual block. Here $\mathbf{h}_i$ is the i^{th} 'hidden' tensor in the network. While technically each layer produces a new tensor, for clarity only the tensors between residual blocks are named $\mathbf{h}_i$. The convolutions inside each respective ResBlock have the same size convolution kernel, as to operate in the same receptive fields. In order to maintain feature size, the input to each convolution is padded using a reflective method. That is, the feature is expanded using data from just inside the boundary.

The full neural network is comprised of n ResBlock layers sandwiched between two convolutions. The 128×128 GP window is input into the additive enhancement network. This input is first convolved and is batch-normalized [97]. Next, a series of n ResBlocks are computed and the output of these layers are summed with the batch-normalized convolved input tensor. Finally, this tensor is put through the final convolutional layer which generates the predicted residual, $\hat{\mathbf{r}}_k \approx \mathbf{y}_k - \mathbf{x}_k$. Figure 4.8 is a architecture diagram illustrating the single image super resolution pipeline, including the additive enhancement network and how it

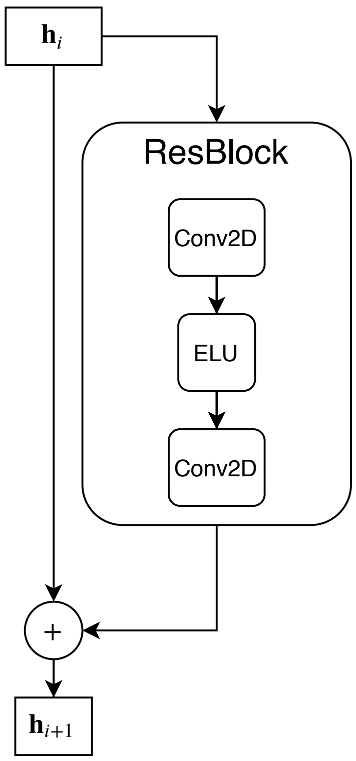


Figure 4.7: The residual block prevalent in the proposed neural network architecture.

is utilized.

While Figure 4.8 illustrates the structure of the network, it does not give details on the hyper-parameter choices. In the method presented, the additive enhancement network contains 6 residual blocks, with decreasing convolution kernel size as the network progresses through each ResBlock. The input layer, and first ResBlock utilize 7×7 sized convolutional kernels. The second and third ResBlocks use 5×5 as the convolution kernel size, and the final 3 ResBlocks have convolution kernels with 3×3. The output layer, however, uses a 1×1 convolution kernel, which effectively generates a linear combination of the final ResBlock's channels to generate the 3 output channels needed for enhancement. For this application, the number of channels for each convolution in the ResBlocks are all 64. The input convolution expands the channels from 3 to 64, and the output channel combines the 64 channel tensor into the 3 channel residual that is needed for the enhancement. The channel sizes and structures are all hyper-parameters that could be changed to meet specific applications, and for the application of Single Image Super Resolution the selected hyper-parameters work well.

Training the Additive Enhancement Network

The Additive Enhancement Network is a deep neural network that requires some considerations when training. The training dataset is the DIV2K training set, containing 800 2K+ resolution images. These images are then split into 128×128 windows to utilize in training. In addition to this, the dataset is artificially enlarged by including a vertically flipped version of the windows. Since the GP upsampled image and the ground truth contain (to low order) the same information, training on the residual is advantage – as shown in [108].

During training, the network is fed the GP upsampled 128×128 image patches

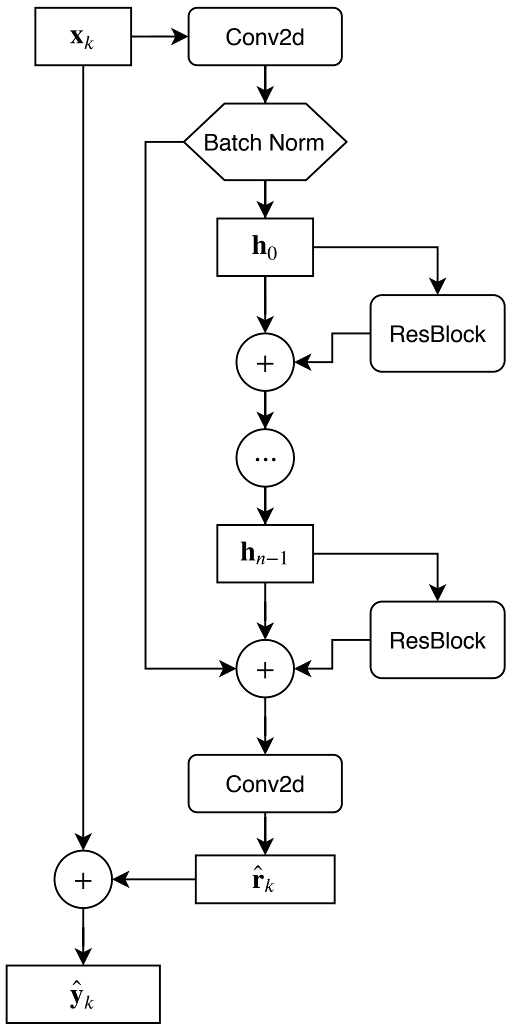


Figure 4.8: Full SISR pipeline with the architecture for the Additive Enhancement Network.

and the output of the network is compared against the true residual – the difference between ground truth's 128×128 window and the corresponding window in the GP upsampled image. The L_1 loss function is utilized for training for both the $2\times$ and $4\times$ cases. Although PSNR is based on the L_2 loss, the L_1 loss can be a better training loss since it is less sensitive to outliers, especially in complex problems like this as seen in [126]. The model weights are initialized by utilizing an orthogonal initializer presented in [170]. This orthogonal initialization is described as an exact solution to problems in deep *linear* neural networks, however, this neural network through the use of batch-normalization and the non-linear activation function ELU is a *non-linear* neural network.

The $4\times$ network was trained over 80 epochs, utilizing a mini-batch of 64 image tiles. The optimizer used was mini-batch stochastic gradient descent, with an initial learning rate of 0.001 and a momentum of 0.9. As the model trained, the learning-rate was decreased by half every 20 epochs. In the $2\times$ case, the network was trained for 43 epochs, slightly over half of number of epochs required in the $4\times$ case using the same training strategy. Training was halted when the model ceased to increase in performance.

Training was performed utilizing 2 NVIDIA GPUs, a Titan XP paired with a GTX 1080Ti. Pairing the GPUs allowed for greater parallelism and reduced the training time. The $4\times$ model took 45 hours to train, and the $2\times$ model took 24 hours.

4.2.3 Results

The performance of the models are compared with a bicubic baseline, the GP upsampling method alone, and very deep convolutional neural network super resolution techniques VDSR proposed in [108] and EDSR illustrated in [126].

This method does not intend to usurp EDSR of its PSNR throne, but to offer a much faster alternative that maintains credible enhancements over upsampling approaches, and inch closer to a viable real-time video super resolution.

In table 4.2, average PSNR, Comparative Sharpness and execution times are presented for the Gaussian Process with Additive Enhancement Network (GPAEN), Bicubic, GP, VDSR, and EDSR. The VDSR method utilizes 20 convolutional layers, and much like GPAEN uses 64 channels depths and trains on the residual between an upsampled image and the ground truth. The EDSR method is another very deep neural network that is based off of ResNet [87], and contains 36 ResBlocks. The proposed method and EDSR have similar ResBlock structures, only EDSR utilizes ReLU [146] and includes a residual scaling layer at the end of each ResBlock. The GPAEN model utilizes a less complex architecture, having 30 fewer ResBlocks than EDSR. Another difference between EDSR and GPAEN is that with EDSR the upsampling routine is embedded into network architecture following the ResBlocks, utilizing more convolutional layers, and a layer that reorganizes feature maps into an image called a pixel shuffle layer.

As represented in table 4.2, GPAEN gives the best performance in PSNR and comparative sharpness per execution time of the Neural Network based approaches. Although EDSR gives a higher PSNR and slightly higher Comparative Sharpness, it is over 140$\times$ slower than GPAEN in the 2$\times$ upsampling context, and 46$\times$ slower in the 4$\times$ context. So for super-resolving a single image, one may prefer the higher performance of EDSR, but for large amounts of image data (entire directories) or video GPAEN will be a much faster solution.

One way to increase the PSNR performance of GPAEN is to utilize the geometric ensemble technique illustrated in [199]. Geometric ensemble is applied for GPAEN as follows, the low resolution image is upsampled utilizing GP, and then

Table 4.2: Mean Peak Signal to Noise Ratio, Comparative Sharpness for 2x and 4x, for GP, GPAEN and other state-of the art techniques over the DIV2K validation set. Average super resolution times for Bicubic, and GP baselines are CPU times, whereas the neural network based approaches are GPU times. The results for VSDR and EDSR are acquired from a publicly available implementations.

	PSNR/CSharp 2x	**PSNR/CSharp 4x**	**Time 2x**	**Time 4x**
Bicubic	31.24dB /0.1635	26.81dB/ 0.0254	0.045s	0.022s
GP	31.31dB/ 0.1964	26.79dB/ 0.0455	0.043s	0.032s
VDSR	32.17dB/ 0.2178	27.21dB/ 0.0592	1.01s	1.00s
GPAEN	33.03dB/ 0.4334	27.90dB/ 0.1020	0.56s	0.55s
GPAENG	33.46dB/ 0.4328	28.22dB/ 0.1024	3.98s	3.95s
EDSR	34.61dB/ 0.4501	28.95dB/ 0.2012	71.09s	23.13s

the GP upsampled image is transformed following the permutations in the Dihedral Group 4 – that is, three 90 degree rotations and corresponding flips. This yields 8 "new" images (including the regular image) that are passed to the Additive Enhancement Network. Following enhancement, the transformed images are inversely transformed to yield images that are justified in the same manner as the original. Next these images are averaged to yield a new super-resolved image. In table 4.2, GPAENG is GPAEN with geometric ensemble. Notice that there is an increase in PSNR in both upsampling cases, but in the $2\times$ case, the comparative sharpness decreases slightly. Utilizing geometric ensemble also increases the computation time by an approximate factor of 8, or the number of transformations applied.

In figures 4.9, 4.10 and 4.11, representative images from the validation set are compared utilizing the base line bicubic method, GP standalone, VDSR, GPAEN and GPAENG.

Figure 4.9, is of two surfers walking to the ocean from some hills. The top image is of the full ground truth. The to better see the comparison between these $2\times$ Super Resolution methods, a zoomed section around the surfer with the cyan wetsuit is examined. The bicubic upsampled image has the most blur and the GP upsampled image is slightly sharper. However, these two baseline upsampling methods do not out perform the neural network enhanced approaches. The bottom row of images contain the VDSR [108] image, and the two GPAEN methods. The two GPAEN images are the closest to the ground truth crop (shown center left) and have better detail in grass next to the surfer than VDSR. Although GPAENG has a higher PSNR value over GPAEN (33.70dB vs 33.75dB respectively) it is not easily noticeable in these windows.

A similar analysis is made in Figure 4.10 comparing again the 2× super resolution methods. This figure contains image 0821 of the DIV2K dataset, which is of a stained glass window. This image has many features that can be difficult to recreate when downsampled and the differences between methods are more subtle. In the ground truth crop (center left), there are 3 black dots in the blue pane of glass near the orange and yellow panes. The GPAEN methods best recreate the intensity of the dark colors in the dots.

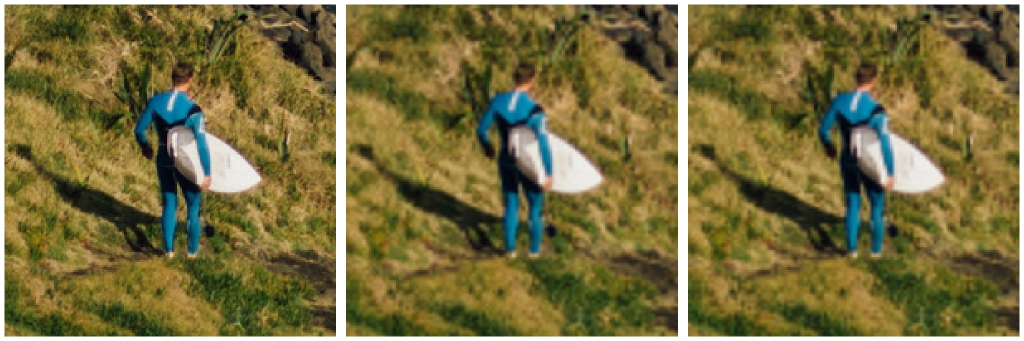
In Figure 4.11, the methods are compared when performing 4× super resolution. This figure is of image 0864, a picture of the Colosseum, the Ancient Roman stadium. The image produced by VDSR smoothes out some of the vertical lines in the ridge below the base of the pillars (bottom left), whereas these lines are better preserved with the GPAEN methods. Additionally, VDSR tends to smooth out the textures on the base of the pillars and the shape is not well reconstructed. When compared to the bicubic method, the base GP upsampling seems to better recreate the shadows on the base of the pillars (center and center right). In all images, the boundary between the first and second pillars (nearest to the sky) is somewhat blended. This blending is seen less in with GPAEN and GPAENG.

A direct comparison between GPAEN and EDSR is presented in figures 4.12

and 4.13 for the 4×. These figures illustrate that the gain in PSNR garnered by EDSR is not easily noticeable by eye. In image 0862, shown in Figure 4.12, the GPAEN reconstructed version gains most of the textures of the fox's fur, present in the ground truth. The EDSR version has a slightly more noticeable snow flakes near the fox's eye.

In Figure 4.13, the GPAEN reconstruction of image 0877 gives qualitatively similar results to the EDSR reconstruction. In the EDSR image the leaves above the flower have a slightly more well defined edge. The colors in the EDSR image are more smoothed out than in the GPAENG image, which gives it a slightly more clear look.

(a) Full Ground Truth Image 0811, with crop bounding box.


(b) Left: Ground Truth, Center: Bicubic, Right: GP

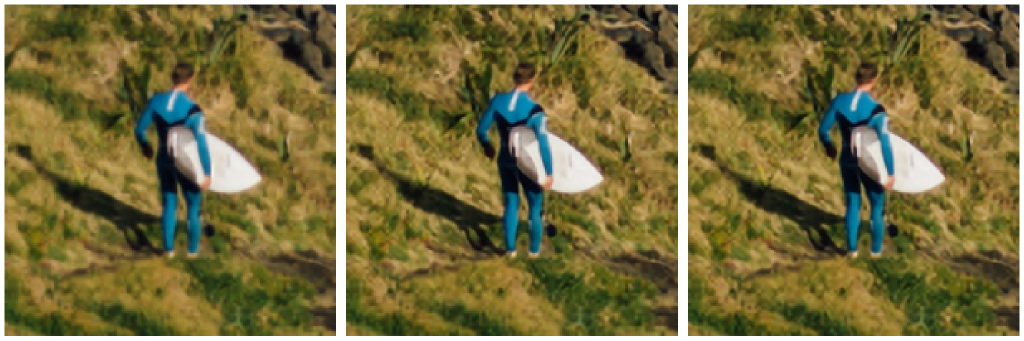
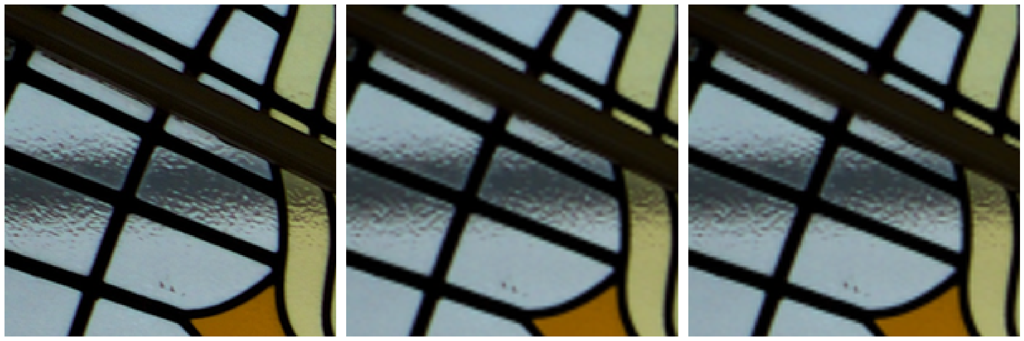

(c) Left: VDSR, Center: GPAEN, Left: GPAENG

Figure 4.9: Visual comparison of 2× SR methods for image 0811.

(a) Full Ground Truth Image 0821, with crop bounding box.


(b) Left: Ground Truth, Center: Bicubic, Right: GP

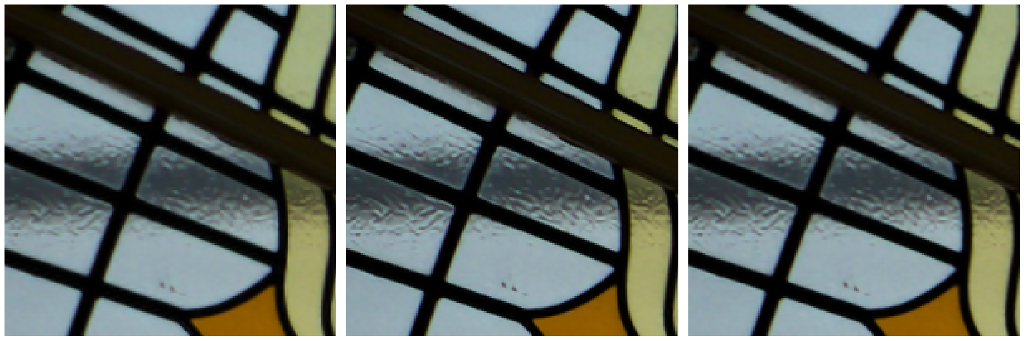
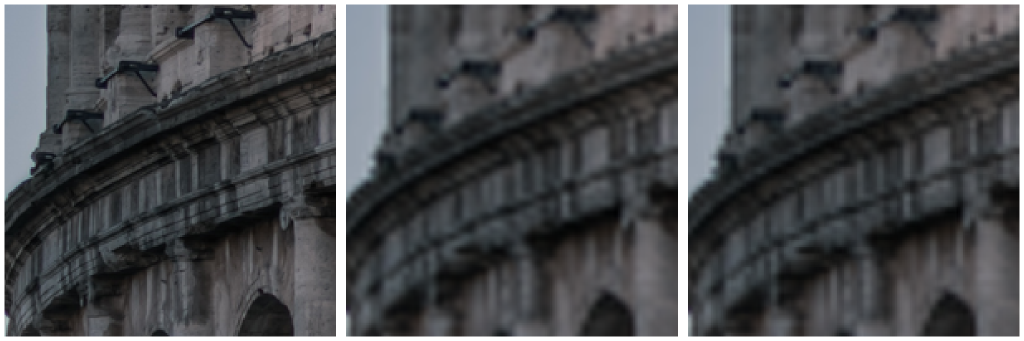
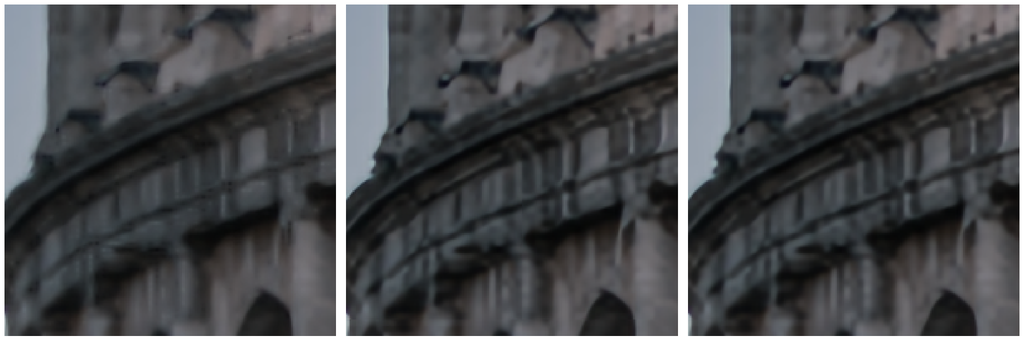

(c) Left: VDSR, Center: GPAEN, Left: GPAENG

Figure 4.10: Visual comparison of 2× SR methods for image 0821.

(a) Full Ground Truth Image 0864, with crop bounding box.

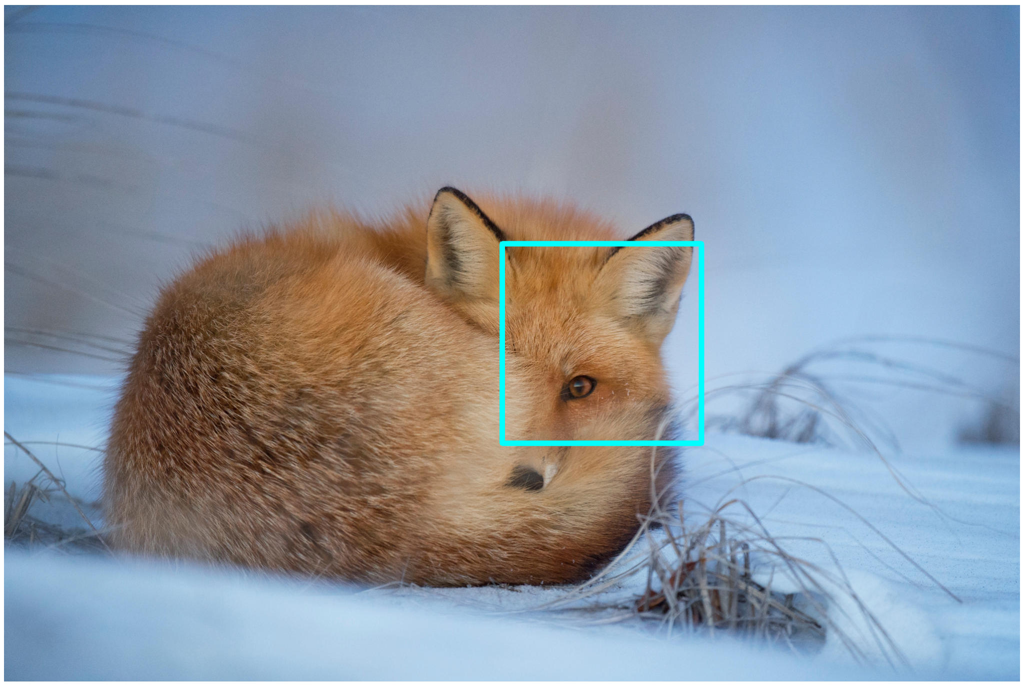


(b) Left: Ground Truth, Center: Bicubic, Right: GP

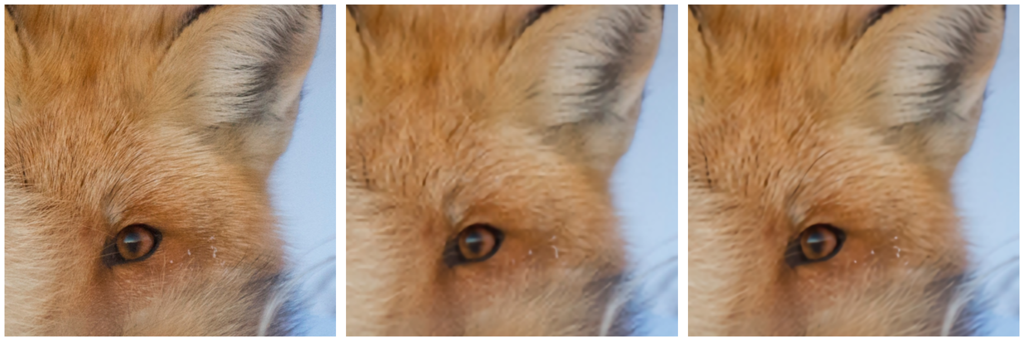


(c) Left: VDSR, Center: GPAEN, Left: GPAENG

Figure 4.11: Visual comparison of 4× SR methods for image 0864.

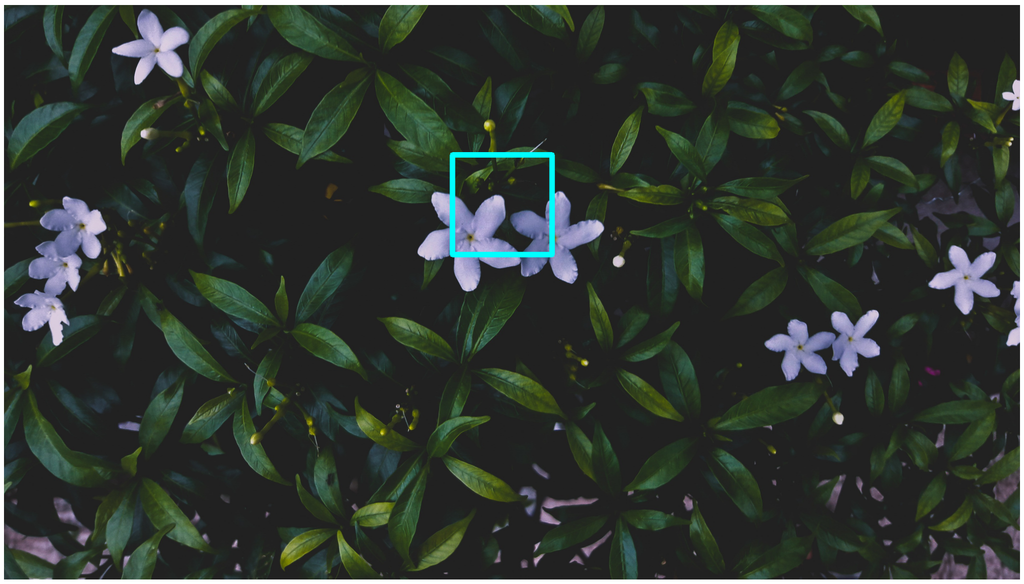


(a) Ground Truth Image 0862 with cropped bounding box.

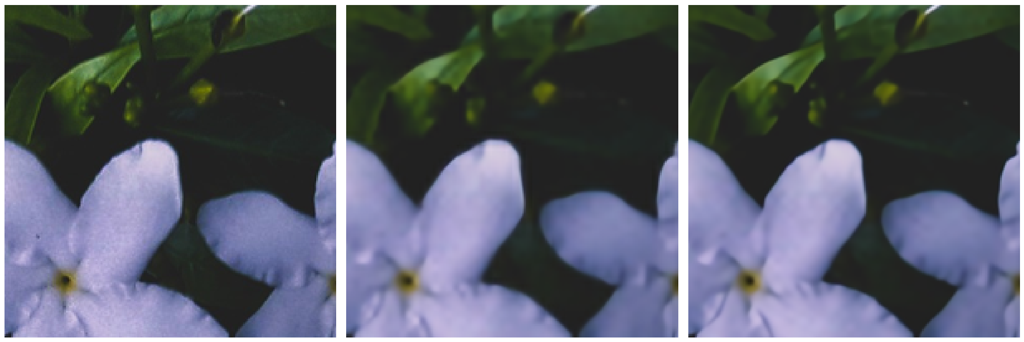


(b) Left: HR crop (PSNR). Center: GPAENG (32.94 dB), Right EDSR (33.30dB)

Figure 4.12: Visual comparison of GPAENG and EDSR for 4× Super Resolution on Image 0862.



(a) Ground Truth Image 0877 with cropped bounding box.



(b) Left: HR crop (PSNR). Center: GPAENG (39.07 dB), Right EDSR (41.14dB)

Figure 4.13: Visual comparison of GPAENG and EDSR for 4× Super Resolution on Image 0877.

4.3 Chapter Summary

The super resolution algorithm illustrated in this chapter provides a high quality relative light weight alternative to EDSR. Through the coupling of a new Gaussian Process based upsampling algorithm and the Additive Enhancement Network, the GPAEN super resolution algorithm gives a high quality images at a fraction of the computational cost of EDSR [126]. Furthermore the algorithm is faster and less complex than VDSR [108] while giving superior results. Additionally, the GP upsampling algorithm gives the user more control over the upsampling. One can decide on a proper length scale and can compute insightful statistics about the dataset. Where as the upsampling portion of EDSR is inextricably tied to the training data and behaves more like a black box.

While GPAEN is still too computationally heavy for realtime video applications, it is fast enough to be used to super resolve large amounts of image data offline or on demand. The new upsampling method without additive enhancement, GP, gives superior comparative sharpness than bicubic while retaining the computational efficiency, and can be used as an alternative for real-time video upsampling. Furthermore the GP upsampling algorithm is embarrassingly parallel, and effectively becomes the process of applying 4 or 16 convolutions for the $2\times$ and $4\times$ upsampling ratios. For fixed resolution inputs, the kernel weights can be computed a-priori and used for every image in a dataset, or frame in a video.

Chapter 5

Conclusion

In this dissertation three different applications of Gaussian Process modeling were described. The first application was to increase the accuracy in Computational Fluid Dynamics simulations that utilize Adaptive Mesh Refinement. The second GP application was to address accuracy issues in low resolution images when used as inputs for Optical Character Recognition. Finally, the last application was for use in a hybrid GP Convolutional Neural Network algorithm to generate high resolution/high fidelity images from low resolution inputs.

The first application will enable scientists to further use higher order reconstruction algorithms while utilizing higher order AMR prolongation in any dimension. Any degradation that could have been caused by linear prolongation is avoided when utilizing higher order reconstruction methods like PPM [40], WENO [183], and the GP methods proposed by Reyes et al [158, 160].

However, this increase in accuracy comes at a computational cost. To maintain conservation of mass, momentum and energy, a multi-modeled approached based on the WENO formulation. However, in this formulation, correct calculation requires $2D+1$ stencils of size $2D+1$. To avoid potential unwanted increase in computation time, an adaptive algorithm was created using a normalized log-likelihood of the center-most stencil S_1. This allows a less computationally complex algorithm to be used in regions of more laminar flow. These regions correspond to a high likelihood that the sampled data points come from a Gaussian Process generated using the Squared Exponential Kernel, meaning the points come from a smooth function.

Furthermore when the algorithm is implemented in a highly parallelized fashion, the computational complexity is distributed and the time cost is mitigated. With the coupling of the parallel computing and the adaptive algorithm, execution time becomes on par with the parallel implementation of the linear interpolation. With this implementation, scientists can choose between computational performance and numerical stability in highly non-laminar flow.

For the application of upsampling for optical character recognition (OCR), low resolution document images were generated to test a 5×5 pixel patch Gaussian Process upsampling model. In contrast to the GP algorithms for AMR, the covariance kernel selected was not based on the Squared Exponential function. Instead the Matèrn 3/2 covariance function was utilized for its superior performance on discontinuous data.

The algorithm focuses on single-channel grayscale document images to improve edges and features used for character recognition. Using this algorithm, tesseract [189] OCR relative accuracy increases by nearly 200% against low resolution images, and increases 6.26% over the bicubic method. In addition to the raw OCR accuracy increase, variance in OCR accuracy decreased, and the minimum and maximum accuracies also increased over the LRDE dataset [113].

The goal of this algorithm was to generate a more accurate representation of the document image while still being computationally efficient in order to be effectively used in production OCR pipelines.

The final application is a generalization of the image upsampling algorithm found in the previous application. The GP upsampling algorithm has a reduction in the sample space bringing it down to a 3×3 pixel window instead of the OCR algorithms 5×5. Additionally, the use of the maximum likelihood estimation is dropped. The inclusion of the MLE causes constant samples to be interpolated as constant, which is advantageous in the context of pixel representation of written characters. In the case of a more generalized super-resolution, the regularization that the maximum likelihood estimator brings may duff out textures that are present in the actual representation, and is more computationally costly. Additionally, the use of the GP algorithm to upsample the can be seen as an initial condition to produce a true high fidelity representation of the ground truth image. This GP algorithm has shown to better at reconstructing sharpness in images over the bicubic baseline algorithm, while maintaining similar or better peak signal to noise ratios.

In order to bridge the gap between state-of-the-art deep learning methods and computationally efficient interpolations, the Additive Enhancement Network is designed to be effective as well as efficient when paired with GP upsampling. The architecture takes generates a simplified variants of the ResBlock of ResNet [87], much like the blocks in EDSR [126], but is paired with an upsampling algorithm like VDSR [108]. Contrary to the previously mentioned deep neural networks, GPAEN is several layers shallower than VDSR and has 30 fewer ResBlocks than EDSR. Additionally, the classic ReLU activation function is swapped out for ELU in order to address vanishing gradient problems prevalent in deep neural networks. Furthermore, the GPAEN architecture is modular, so one could extend GPAEN to be deeper, or shallower to match one's requirements.

Together the Additive Enhancement Network and the Gaussian Process upsampling algorithm form a fast and high quality SISR model. The GPAEN method yields better performance than VDSR with half the computation cost, and achieves much of the raw performance EDSR garners without the 70s average compute time.

This dissertation further illustrates the power and applicability of Gaussian Processes for general interpolation purposes. Furthermore, pairing GP along with additional models can solve a breadth of problems, no matter how seemingly unrelated they appear to be.

Bibliography

[1] A direct Arbitrary-Lagrangian–Eulerian ADER-WENO finite volume scheme on unstructured tetrahedral meshes for conservative and non-conservative hyperbolic systems in 3d. *Journal of Computational Physics*, 275:484–523, October 2014.

[2] *Proceedings of International Conference on Computer Vision*, 2017.

[3] *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2018.

[4] M Adams, PO Schwartz, H Johansen, P Colella, TJ Ligocki, D Martin, et al. Chombo software package for amr applications-design document (tech. rep.). *Berkeley, CA: Applied Numerical Algorithms Group Computational Research Division Lawrence Berkeley National Laboratory*, 2015.

[5] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[6] A. S. Almgren. Introduction to block structured adaptive mesh refinement (amr). In *HIPACC*, 2011.

[7] A. S. Almgren, V. E. Beckner, J.B Bell, M. Day, L. Howell, Joggerst C., M.J. Lijewski, A. Nonaka, M. Singer, and M. Zingale. Castro: A new compressible astrophysical solver. i. hydrodynamics and self-gravity. *The Astrophysical Jounral*, 715:1221–1238, 2010.

[8] A. S. Almgren, J. B. Bell, P. Collela, L. H. Howell, and M. L. Welcome. A conservative adaptive projection method for the variable density incompressible navier-stokes equations. *Journal of Computationl Physics*, 142:1–46, 1998.

[9] Ann S Almgren, Vincent E Beckner, John B Bell, MS Day, Louis H Howell, CC Joggerst, MJ Lijewski, Andy Nonaka, M Singer, and Michael Zingale. Castro: A new compressible astrophysical solver. i. hydrodynamics and self-gravity. *The Astrophysical Journal*, 715(2):1221, 2010.

[10] Muhammed Atak, Andrea Beck, Thomas Bolemann, David Flad, Hannes Frank, Florian Hindenlang, and Claus-Dieter Munz. Discontinuous Galerkin for high performance computational fluid dynamics. In *High Performance Computing in Science and Engineering '14*, pages 499–518. Springer, 2015.

[11] Norbert Attig, Paul Gibbon, and Th Lippert. Trends in supercomputing: The European path to exascale. *Computer Physics Communications*, 182(9):2041–2046, 2011.

[12] Mahboub Baccouch. Asymptotically exact a posteriori local discontinuous Galerkin error estimates for the one-dimensional second-order wave equation. *Numerical Methods for Partial Differential Equations*, 31(5):1461–1491, 2015.

[13] Dinshaw S Balsara. Multidimensional HLLE Riemann solver: Application to euler and magnetohydrodynamic flows. *Journal of Computational Physics*, 229(6):1970–1993, 2010.

[14] Dinshaw S. Balsara. Three dimensional HLL Riemann solver for conservation laws on structured meshes; Application to Euler and magnetohydrodynamic flows. *Journal of Computational Physics*, 295:1–23, August 2015.

[15] Dinshaw S. Balsara. Higher-order accurate space-time schemes for computational astrophysics—Part I: finite volume methods. *Living Reviews in Computational Astrophysics*, 3(1):2, 2017.

[16] Dinshaw S. Balsara, Sudip Garain, and Chi Wang Shu. An efficient class of WENO schemes with adaptive order. *Journal of Computational Physics*, 326:780–804, 2016.

[17] Dinshaw S Balsara, Chad Meyer, Michael Dumbser, Huijing Du, and Zhiliang Xu. Efficient implementation of ADER schemes for Euler and magnetohydrodynamical flows on structured meshes–speed comparisons with Runge-Kutta methods. *Journal of Computational Physics*, 235:934–969, 2013.

[18] Dinshaw S Balsara, Tobias Rumpf, Michael Dumbser, and Claus-Dieter Munz. Efficient, high accuracy ADER-WENO schemes for hydrodynamics and divergence-free magnetohydrodynamics. *Journal of Computational Physics*, 228(7):2480–2516, 2009.

[19] Andrea D Beck, Thomas Bolemann, David Flad, Hannes Frank, Gregor J Gassner, Florian Hindenlang, and Claus-Dieter Munz. High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations. *International Journal for Numerical Methods in Fluids*, 76(8):522–548, 2014.

[20] J. Bell, M. Berger, Saltzman J., and M. Welcome. A three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM Journal of Scientific Computing*, 15:127–138, 1994.

[21] M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82:64–84, 1989.

[22] M.J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:484–512, 1984.

[23] Caterina Bigoni and Jan S. Hesthaven. Adaptive WENO methods based on radial basis function reconstruction. *Journal of Scientific Computing*, 2017.

[24] C Bishop. *Pattern recognition and machine learning (information science and statistics).* Springer, New York, New York, first edition, 2006.

[25] Johan Bjorck, Carla P. Gomes, and Bart Selman. Understanding batch normalization. *CoRR*, abs/1806.02375, 2018.

[26] Rafael Borges, Monique Carmona, Bruno Costa, and Wai Sun Don. An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. *Journal of Computational Physics*, 227(6):3191–3211, 2008.

[27] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[28] Greg L Bryan, Michael L Norman, Brian W O'Shea, Tom Abel, John H Wise, Matthew J Turk, Daniel R Reynolds, David C Collins, Peng Wang, Samuel W Skillman, et al. Enzo: An adaptive mesh refinement code for astrophysics. *The Astrophysical Journal Supplement Series*, 211(2):19, 2014.

[29] Pawel Buchmüller and Christiane Helzel. Improved accuracy of high-order WENO finite volume methods on Cartesian grids. *Journal of Scientific Computing*, 61(2):343–368, 2014.

[30] Martin D Buhmann. Radial basis functions. *Acta Numerica*, 9:1–38, 2000.

[31] Waixiang Cao, Chi-Wang Shu, Yang Yang, and Zhimin Zhang. Superconvergence of discontinuous Galerkin methods for two-dimensional hyperbolic equations. *SIAM Journal on Numerical Analysis*, 53(4):1651–1671, 2015.

[32] Cristóbal E Castro. High order ADER FV/DG numerical methods for hyperbolic equations. *Monographs of the School of Doctoral Studies in Environmental Engineering. Trento, Italy: University of Trento*, 2007.

[33] Yanlai Chen, Sigal Gottlieb, Alfa Heryudono, and Akil Narayan. A reduced radial basis function method for partial differential equations on irregular domains. *Journal of Scientific Computing*, 66(1):67–90, 2016.

[34] Yuxi Chen, Gábor Tóth, and Tamas I. Gombosi. A fifth-order finite difference scheme for hyperbolic equations on block-adaptive curvilinear grids. *Journal of Computational Physics*, 305:604–621, 2016.

[35] Zexun Chen, Bo Wang, and Alexander N. Gorban. Multivariate gaussian and student$-t$ process regression for multi-output prediction, 2017.

[36] Bernardo Cockburn, Fengyan Li, and Chi-Wang Shu. Locally divergence-free discontinuous Galerkin methods for the Maxwell equations. *Journal of Computational Physics*, 194(2):588–610, 2004.

[37] Bernardo Cockburn and Chi-Wang Shu. The Runge-Kutta discontinuous Galerkin method for conservation laws V: Multidimensional systems. *Journal of Computational Physics*, 141(2):199–224, 1998.

[38] Bernardo Cockburn and Chi-Wang Shu. Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. *Journal of Scientific Computing*, 16(3):173–261, 2001.

[39] P Colella, DT Graves, TJ Ligocki, DF Martin, D Modiano, DB Serafini, and B Van Straalen. Chombo software package for amr applications design document. *Available at the Chombo website: http://seesar. lbl. gov/ANAG/-chombo/(September 2008)*, 2009.

[40] P Colella and P. R. Woodward. The piecewise parabolic method (ppm) for gas-dynamical simulations. *Journal of Computational Physics*, 54:174–201, 1984.

[41] Phillip Colella, Milo R. Dorr, Jeffrey A. F. Hittinger, and Daniel F. Martin. High-order, finite-volume methods in mapped coordinates. *J. Comput. Physics*, 230:2952–2976, 2010.

[42] Phillip Colella and Paul R Woodward. The piecewise parabolic method (PPM) for gas-dynamical simulations. *Journal of Computational Physics*, 54(1):174–201, 1984.

[43] P. Collela and H. Glaz. Efficient solution algorithms for the riemann problem for real gases. *Journal of Computational Physics*, 59:264–289, 1985.

[44] Isabella Cravero and Matteo Semplice. On the accuracy of WENO and CWENO reconstructions of third order on nonuniform meshes. *Journal of Scientific Computing*, 67(3):1219–1246, 2016.

[45] Andrew J Cunningham, Adam Frank, Peggy Varnière, Sorin Mitran, and Thomas W Jones. Simulating magnetohydrodynamical flow with constrained transport and adaptive mesh refinement: algorithms and tests of the astrobear code. *The Astrophysical Journal Supplement Series*, 182(2):519, 2009.

[46] Josep de la Puente, Martin Käser, Michael Dumbser, and Heiner Igel. An arbitrary high-order discontinuous galerkin method for elastic waves on unstructured meshes-iv. anisotropy. *Geophysical Journal International*, 169(3):1210–1228, 2007.

[47] L. Del Zanna, N. Bucciantini, and P. Londrillo. An efficient shock-capturing central-type scheme for multidimensional relativistic flows: II. Magnetohydrodynamics. *Astronomy & Astrophysics*, 400(2):397–413, March 2003.

[48] L. Del Zanna, O. Zanotti, N. Bucciantini, and P. Londrillo. ECHO: a Eulerian conservative high-order scheme for general relativistic magnetohydrodynamics and magnetodynamics. *Astronomy & Astrophysics*, 473(1):11–30, 2007.

[49] Peter Delmont, Rony Keppens, and Bart van der Holst. An exact Riemann-solver-based solution for regular shock refraction. *Journal of Fluid Mechanics*, 627:33–53, 2009.

[50] Xiaogang Deng and Hanxin Zhang. Developing high-order weighted compact nonlinear schemes. *Journal of Computational Physics*, 165(1):22–44, 2000.

[51] Wai-Sun Don, Zhen Gao, Peng Li, and Xiao Wen. Hybrid compact-WENO finite difference scheme with conjugate Fourier shock detection algorithm for hyperbolic conservation laws. *SIAM Journal on Scientific Computing*, 38(2):A691–A711, January 2016.

[52] C. Dong and C. Change Loy. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38, February 2016.

[53] C. Dong, C. Change Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *Computer Vision – ECCV 2014*, pages 184–199. Springer International Publishing, 2014.

[54] JJ Dongarra. *On the Future of High Performance Computing: How to Think for Peta and Exascale Computing.* Hong Kong University of Science and Technology, 2012.

[55] Anshu Dubey, Ann Almgren, John Bell, Martin Berzins, Steve Brandt, Greg Bryan, Phillip Colella, Daniel Graves, Michael Lijewski, Frank Löffler, et al. A survey of high level frameworks in block-structured adaptive mesh refinement packages. *Journal of Parallel and Distributed Computing*, 74(12):3217–3227, 2014.

[56] Anshu Dubey, Katie Antypas, Murali K Ganapathy, Lynn B Reid, Katherine Riley, Dan Sheeler, Andrew Siegel, and Klaus Weide. Extensible component-based architecture for flash, a massively parallel, multiphysics simulation code. *Parallel Computing*, 35(10-11):512–522, 2009.

[57] M. Dumbser, D. S. Balsara, E. F. Toro, and C.-D. Munz. A unified framework for the construction of one-step finite volume and discontinuous Galerkin schemes on unstructured meshes. *Journal of Computational Physics*, 227:8209–8253, September 2008.

[58] Michael Dumbser and Martin Käser. An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes—II. The three-dimensional isotropic case. *Geophysical Journal International*, 167(1):319–336, 2006.

[59] Michael Dumbser, Martin Käser, Vladimir A Titarev, and Eleuterio F Toro. Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems. *Journal of Computational Physics*, 226(1):204–243, 2007.

[60] Michael Dumbser, Martin Kaser, and Eleuterio F Toro. An arbitrary high-order discontinuous galerkin method for elastic waves on unstructured meshes-v. local time stepping and p-adaptivity. *Geophysical Journal International*, 171(2):695–717, 2007.

[61] Michael Dumbser and CD Munz. Arbitrary high order discontinuous galerkin schemes. *Numerical methods for hyperbolic and kinetic problems*, pages 295–333, 2005.

[62] Michael Dumbser and Claus-Dieter Munz. ADER discontinuous Galerkin schemes for aeroacoustics. *Comptes Rendus Mécanique*, 333(9):683–687, 2005.

[63] Michael Dumbser and Claus-Dieter Munz. Building blocks for arbitrary high order discontinuous Galerkin schemes. *Journal of Scientific Computing*, 27(1-3):215–230, 2006.

[64] Michael Dumbser, Olindo Zanotti, Arturo Hidalgo, and Dinshaw S Balsara. Ader-weno finite volume schemes with space–time adaptive mesh refinement. *Journal of Computational Physics*, 248:257–286, 2013.

[65] Deepak Eachempati, Hyoung Joon Jun, and Barbara Chapman. An open-source compiler and runtime implementation for Coarray Fortran. In *Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model*, PGAS '10, pages 13:1–13:8, New York, NY, USA, 2010. ACM.

[66] M. Emmett, E. Motheau, W. Zhang, M. Minion, and J.B Bell. A fourth-order adaptive mesh refinement algorithm for the multicomponent, reacting compressible navier-stokes equations. *Combustion Theory and Modeling*, 2019.

[67] Christoph Feichtenhofer, Hannes Fassold, and Peter Schallauer. A perceptual image sharpness metric based on local edge gradient analysis. *IEEE Signal Processing Letters*, 20:379–382, 2013.

[68] Jared O Ferguson, Christiane Jablonowski, Hans Johansen, Peter McCorquodale, Phillip Colella, and Paul A Ullrich. Analyzing the adaptive mesh refinement (amr) characteristics of a high-order 2d cubed-sphere shallow-water model. *Monthly Weather Review*, 144(12):4641–4666, 2016.

[69] Richard Franke. Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, 38(157):181–200, 1982.

[70] B Fryxell, K Olson, P Ricker, FX Timmes, M Zingale, DQ Lamb, P MacNeice, R Rosner, JW Truran, and H Tufo. FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *The Astrophysical Journal Supplement Series*, 131(1):273, 2000.

[71] Sudip Garain, Dinshaw S. Balsara, and John Reid. Comparing Coarray Fortran (CAF) with MPI for several structured mesh PDE applications. *Journal of Computational Physics*, 297:237–253, September 2015.

[72] F.A Gers. Learning to forget: continual prediction with lstm. *9th International Conference on Artificial Neural Networks: ICANN*, 1999.

[73] Gary A Glatzmaiers and Paul H Roberts. A three-dimensional self-consistent computer simulation of a geomagnetic field reversal. *Nature*, 377(6546):203–209, 1995.

[74] A Glocer, G Tóth, Y Ma, T Gombosi, J-C Zhang, and LM Kistler. Multifluid block-adaptive-tree solar wind roe-type upwind scheme: Magnetospheric composition and dynamics during geomagnetic storms—initial results. *Journal of Geophysical Research: Space Physics*, 114(A12), 2009.

[75] Sergei Konstantinovich Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, 47(89)(3):271–306, 1959.

[76] Muhammad Waleed Gondal, Bernhard Schölkopf, and Michael Hirsch. The unreasonable effectiveness of texture transfer for single image super-resolution. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision – ECCV 2018 Workshops*, pages 80–97, Cham, 2019. Springer International Publishing.

[77] Sigal Gottlieb, David Ketcheson, and Chi-Wang Shu. *Strong stability preserving Runge-Kutta and multistep time discretizations.* 2011.

[78] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, Bunke H., and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition, 2008.

[79] Jingyang Guo and Jae-Hun Jung. Non-polynomial ENO and WENO finite volume methods for hyperbolic conservation laws. m:1–23, 2016.

[80] Jingyang Guo and Jae-Hun Jung. A RBF-WENO finite volume method for hyperbolic conservation laws with the monotone polynomial interpolation method. *Applied Numerical Mathematics*, 112:27–50, 2017.

[81] Xiaocheng Guo, Vladimir Florinski, and Chi Wang. The HLLD riemann solver based on magnetic field decomposition method for the numerical simulation of magneto-hydrodynamics. *Journal of Computational Physics*, 327:543–552, 2016.

[82] R.A. Haddad and A.N. Akansu. A class of fast gaussian binomial filters for speech and image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 39, 1991.

[83] W. Haijun, X. Gao, K. Zhang, and J. Li. Single image super-resolution using gaussian process regression with dictionary-based sampling and student-t likelihood. *IEEE Transactions of Image Processing*, 26:3556–3568, 2017.

[84] Ami Harten, Bjorn Engquist, Stanley Osher, and Sukumar R Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, III. *Journal of Computational Physics*, 71(2):231–303, 1987.

[85] Amiram Harten, Peter D Lax, and Bram van Leer. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. *SIAM review*, 25(1):35–61, 1983.

[86] H. He and W. Siu. Single image super-resolution using gaussian process regression. *IEEE Xplore*, 2011.

[87] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[88] Andrew K. Henrick, Tariq D. Aslam, and Joseph M. Powers. Mapped weighted essentially non-oscillatory schemes: Achieving optimal order near critical points. *Journal of Computational Physics*, 207(2):542–567, August 2005.

[89] Alfa RH Heryudono and Tobin A Driscoll. Radial basis function interpolation on irregular domain through conformal transplantation. *Journal of Scientific Computing*, 44(3):286–300, 2010.

[90] Jan S Hesthaven, Sigal Gottlieb, and David Gottlieb. *Spectral methods for time-dependent problems*, volume 21. Cambridge University Press, 2007.

[91] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9, 1997.

[92] Richard D Hornung and Scott R Kohn. Managing application complexity in the samrai object-oriented framework. *Concurrency and computation: practice and experience*, 14(5):347–368, 2002.

[93] https://pelec.readthedocs.io/en/latest/. Pelec, an compressible, adaptive mesh, combustion code.

[94] Benyamin Jafari (https://stats.stackexchange.com/users/209206/benyamin jafari). Difference between feedback rnn and lstm/ gru. Cross Validated. URL:https://stats.stackexchange.com/q/420172 (version: 2019-08-14).

[95] Z. Hui, X. Wang, and X. Goa. Fast and accurate single image super-resolution via information distillation network. In *Proceedings of Conference on Computer Vision and Pattern Recognition* [3].

[96] J. Smith III. Spectral audio signal processing. 2011.

[97] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[98] Leland Jameson. Amr vs high order schemes. *Journal of Scientific Computing*, 18(1):1–24, 2003.

[99] Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126(1):202–228, 1996.

[100] Yan Jiang, Chi-Wang Shu, and Mengping Zhang. An alternative formulation of finite difference weighted ENO schemes with Lax–Wendroff time discretization for conservation laws. *SIAM Journal on Scientific Computing*, 35(2):A1137–A1160, January 2013.

[101] George C Jordan IV, RT Fisher, DM Townsley, AC Calder, C Graziani, S Asida, DQ Lamb, and JW Truran. Three-dimensional simulations of the deflagration phase of the gravitationally confined detonation model of type ia supernovae. *The Astrophysical Journal*, 681(2):1448, 2008.

[102] Martin Käser and Michael Dumbser. An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes—I. The two-dimensional isotropic case with external source terms. *Geophysical Journal International*, 166(2):855–877, 2006.

[103] Martin Käser, Michael Dumbser, Josep De La Puente, and Heiner Igel. An arbitrary high-order discontinuous galerkin method for elastic waves on unstructured meshes—iii. viscoelastic attenuation. *Geophysical Journal International*, 168(1):224–242, 2007.

[104] Aaron Katz and Antony Jameson. A comparison of various meshless schemes within a unified algorithm. *AIAA paper*, 594, 2009.

[105] Rony Keppens, M Nool, G Tóth, and JP Goedbloed. Adaptive mesh refinement for conservative systems: multi-dimensional efficiency evaluation. *Computer Physics Communications*, 153(3):317–339, 2003.

[106] R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29, 1981.

[107] Alexei M Khokhlov. Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations. *Journal of Computational Physics*, 143(2):519–543, 1998.

[108] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR Oral)*, June 2016.

[109] Richard I Klein. Star formation with 3-d adaptive mesh refinement: the collapse and fragmentation of molecular clouds. *Journal of Computational and Applied Mathematics*, 109(1-2):123–152, 1999.

[110] Andreas Klöckner, Tim Warburton, Jeff Bridge, and Jan S Hesthaven. Nodal discontinuous Galerkin methods on graphics processors. *Journal of Computational Physics*, 228(21):7863–7882, 2009.

[111] Andrey V Kravtsov, Anatoly A Klypin, and Alexei M Khokhlov. Adaptive refinement tree: a new high-resolution n-body code for cosmological simulations. *The Astrophysical Journal Supplement Series*, 111(1):73, 1997.

[112] Peter D Lax and Xu-Dong Liu. Solution of two-dimensional Riemann problems of gas dynamics by positive schemes. *SIAM Journal on Scientific Computing*, 19(2):319–340, 1998.

[113] Guillaume Lazzara, Roland Levillain, Thierry Géraud, Yann Jacquelet, Julien Marquegnies, and Arthur Crepin-Leblond. The scribo module of the olena platform: A free software framework for document image analysis. *2011 International Conference on Document Analysis and Recognition*, pages 252–258, 2011.

[114] D. Lecoanet, M. McCourt, E. Quataert, K. J. Burns, G. M. Vasil, J. S. Oishi, B. P. Brown, J. M. Stone, and R. M. O'Leary. A validated nonlinear Kelvin-Helmholtz benchmark for numerical hydrodynamics. *mnras*, 455:4274–4288, February 2016.

[115] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[116] Dongwook Lee, Hugues Faller, and Adam Reyes. The piecewise cubic method (PCM) for computational fluid dynamics. *Journal of Computational Physics*, 341:230–257, 2017.

[117] Luis Lehner, Steven L Liebling, and Oscar Reula. Amr, stability and higher accuracy. *Classical and Quantum Gravity*, 23(16):S421, 2006.

[118] Sanjiva K Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, 1992.

[119] Randall J LeVeque. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002.

[120] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, volume 98. SIAM, 2007.

[121] Doron Levy, Gabriella Puppo, and Giovanni Russo. Compact central WENO schemes for multidimensional conservation laws. *SIAM Journal on Scientific Computing*, 22(2):656–672, 2000.

[122] Fengyan Li and Chi-Wang Shu. Locally divergence-free discontinuous Galerkin methods for MHD equations. *Journal of Scientific Computing*, 22(1-3):413–442, 2005.

[123] Juan Li, Zhenwei Zhao, Andrei Kazakov, and Frederick L. Dryer. An updated comprehensive kinetic model of hydrogen combustion. *International Journal of Chemical Kinetics*, 36(10):566–575, 2004.

[124] Shengtai Li. An HLLC Riemann solver for magneto-hydrodynamics. *Journal of Computational Physics*, 203(1):344–357, 2005.

[125] X. Li and X. Wu. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. *CoRR*, abs/1410.4281, 2014.

[126] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. *CoRR*, abs/1707.02921, 2017.

[127] Xiao-Yan Liu, Andreas Karageorghis, and CS Chen. A Kansa-radial basis function method for elliptic boundary value problems in annular domains. *Journal of Scientific Computing*, 65(3):1240–1269, 2015.

[128] Xu-Dong Liu, Stanley Osher, and Tony Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200–212, 1994.

[129] Yingjie Liu, Chi-Wang Shu, Eitan Tadmor, and Mengping Zhang. L2 stability analysis of the central discontinuous Galerkin method and a comparison between the central and regular discontinuous Galerkin methods. *ESAIM: Mathematical Modelling and Numerical Analysis*, 42(4):593–607, 2008.

[130] P. MacNeice, K.M. Olson, C. Mobarry, R. Fainchtein, and C. Packer. Paramesh: A parallel adaptive mesh refinement community toolkit. *NASA/CR*, 209483, 1999.

[131] Peter MacNeice, Kevin M Olson, Clark Mobarry, Rosalinda De Fainchtein, and Charles Packer. Paramesh: A parallel adaptive mesh refinement community toolkit. *Computer physics communications*, 126(3):330–354, 2000.

[132] Jordan M Martel and Rodrigo B Platte. Stability of radial basis function methods for convection problems on the circle and sphere. *Journal of Scientific Computing*, 69(2):487–505, 2016.

[133] Peter McCorquodale and Phillip Colella. A high-order finite-volume method for conservation laws on locally refined grids. *Communications in Applied Mathematics and Computational Science*, 6(1):1–25, 2011.

[134] Peter McCorquodale, Paul Ullrich, Hans Johansen, and Phillip Colella. An adaptive multiblock high-order finite-volume method for solving the shallow-water equations on the sphere. *Communications in Applied Mathematics and Computational Science*, 10(2):121–162, 2015.

[135] Jena Meinecke, HW Doyle, Francesco Miniati, AR Bell, R Bingham, R Crowston, RP Drake, Milad Fatenejad, Michel Koenig, Yasuhiro Kuramitsu, et al. Turbulent amplification of magnetic fields in laboratory laser-produced shock waves. *Nature Physics*, 10(7):520, 2014.

[136] A. Mignone, P. Tzeferacos, and G. Bodo. High-order conservative finite difference GLM-MHD schemes for cell-centered MHD. *Journal of Computational Physics*, 229(17):5896 – 5920, 2010.

[137] Andrea Mignone, Petros Tzeferacos, and Gianluigi Bodo. High-order conservative finite difference GLM–MHD schemes for cell-centered MHD. *Journal of Computational Physics*, 229(17):5896–5920, 2010.

[138] Andrea Mignone, C Zanni, Petros Tzeferacos, B Van Straalen, P Colella, and G Bodo. The pluto code for adaptive mesh computations in astrophysical fluid dynamics. *The Astrophysical Journal Supplement Series*, 198(1):7, 2011.

[139] Francesco Miniati and Daniel F Martin. Constrained-transport magnetohydrodynamics with adaptive mesh refinement in charm. *The Astrophysical Journal Supplement Series*, 195(1):5, 2011.

[140] M.L. Minion. Semi-implicit spectral deferred correction methods for ordinary differential equations. *Communications in Mathematical Sciences*, 2003.

[141] Takahiro Miyoshi and Kanya Kusano. A multi-state HLL approximate Riemann solver for ideal magnetohydrodynamics. *Journal of Computational Physics*, 208(1):315–344, 2005.

[142] Timothy J Moroney and Ian W Turner. A finite volume method based on radial basis functions for two-dimensional nonlinear diffusion equations. *Applied Mathematical Modelling*, 30(10):1118–1133, 2006.

[143] Timothy J Moroney and Ian W Turner. A three-dimensional finite volume method based on radial basis functions for the accurate computational modelling of nonlinear diffusion equations. *Journal of Computational Physics*, 225(2):1409–1426, 2007.

[144] KW Morton and T. Sonar. Finite volume methods for hyperbolic conservation laws. *Acta Numerica*, 16(1):155–238, 2007.

[145] Andrew Nonaka, AS Almgren, JB Bell, MJ Lijewski, CM Malone, and M Zingale. Maestro: An adaptive low mach number hydrodynamics algorithm for stellar flows. *The Astrophysical Journal Supplement Series*, 188(2):358, 2010.

[146] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *ArXiv*, abs/1811.03378, 2018.

[147] Chirag I. Patel, Atul Patel, and Dharmendra T. Patel. Optical character recognition by open source ocr tool tesseract: A case study. 2012.

[148] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[149] Christian Pelties, Josep Puente, Jean-Paul Ampuero, Gilbert B Brietzke, and Martin Käser. Three-dimensional dynamic rupture simulation with a high-order discontinuous Galerkin method on unstructured tetrahedral meshes. *Journal of Geophysical Research: Solid Earth*, 117(B2), 2012.

[150] Tomasz Plewa, Timur Linde, and V Gregory Weirs. Adaptive mesh refinement-theory and applications. *Lecture notes in computational science and engineering*, 41:3–5, 2005.

[151] Kenneth G Powell, Philip L Roe, Timur J Linde, Tamas I Gombosi, and Darren L De Zeeuw. A solution-adaptive upwind scheme for ideal magnetohydrodynamics. *Journal of Computational Physics*, 154(2):284–309, 1999.

[152] Michael JD Powell. Radial basis funcitionn for multivariable interpolation: A review. In *IMA Conference on Algorithms for the Approximation of Functions ans Data*, pages 143–167. RMCS, 1985.

[153] Jianzhen Qian, Jiequan Li, and Shuanghu Wang. The generalized Riemann problems for compressible fluid flows: Towards high order. *Journal of Computational Physics*, 259:358–389, 2014.

[154] Jianxian Qiu and Chi-Wang Shu. On the construction, comparison, and local characteristic decomposition for high-order central WENO schemes. *Journal of Computational Physics*, 183(1):187–209, November 2002.

[155] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation And Machine Learning. MIT Press, 2005.

[156] Jaideep Ray, Christopher A Kennedy, Sophia Lefantzi, and Habib N Najm. Using high-order methods on adaptively refined block-structured meshes: derivatives, interpolations, and filters. *SIAM Journal on Scientific Computing*, 29(1):139–181, 2007.

[157] William H Reed and TR Hill. Triangular mesh methods for the neutron transport equation. Technical report, Los Alamos Scientific Lab., N. Mex.(USA), 1973.

[158] A. Reyes, D. Lee, Grasiani, C., and Tzeferacors P. A new class of high-order mehtods for fluid dynaimcs simulation using gaussian process modeling. *Journal of Computational Physics*, 2017.

[159] Adam Reyes, Dongwook Lee, Carlo Graziani, and Petros Tzeferacos. A new class of high-order methods for fluid dynamics simulations using Gaussian Process Modeling. *Journal of Scientific Computing*, 76:443–480, 2018.

[160] Adam Reyes, Dongwook Lee, Carlo Graziani, and Petros Tzeferacos. A variable high-order shock-capturing finite difference method with gp-weno. *Journal of Computational Physics*, 381:189–217, Mar 2019.

[161] Philip L Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

[162] A. M. Rogerson and E. Meiburg. A numerical study of the convergence properties of ENO schemes. *Journal of Scientific Computing*, 5(2):151–167, June 1990.

[163] P. Roy, S. Dutta, N. Dey, G. Dey, S. Chakraborty, and R. Ray. Adaptive thresholding: A comparative study. *International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 2014.

[164] Ali Safdari-Vaighani, Elisabeth Larsson, and Alfa Heryudono. Radial basis function methods for the Rosenau equation and other higher order PDEs. *Journal of Scientific Computing*, 75(3):1555–1580, 2018.

[165] M. Sajjadi, B. Schölkopf, and M Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *Proceedings of International Conference on Computer Vision* [2].

[166] H. Sak, Senior A., and F. Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128, 2014.

[167] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization?, 2018.

[168] Scott A Sarra and Edward J Kansa. Multiquadric radial basis function approximation methods for the numerical solution of partial differential equations. *Advances in Computational Mechanics*, 2(2), 2009.

[169] Richard Saurel, Michel Larini, and Jean Claude Loraud. Exact and approximate Riemann solvers for real gases. *Journal of Computational Physics*, 112(1):126–137, 1994.

[170] M. A. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dunamics of learning in deep linear neural networks. *ICLR*, 2013.

[171] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014.

[172] W Schmidt, J Schulz, L Iapichino, F Vazza, and AS Almgren. Influence of adaptive mesh refinement and the hydro solver on shear-induced mass stripping in a minor-merger scenario. *Astronomy and Computing*, 9:49–63, 2015.

[173] Carsten W Schulz-Rinne. Classification of the Riemann problem for two-dimensional gas dynamics. *SIAM Journal on Mathematical Analysis*, 24(1):76–88, 1993.

[174] Carsten W Schulz-Rinne, James P Collins, and Harland M Glaz. Numerical solution of the Riemann problem for two-dimensional gas dynamics. *SIAM Journal on Scientific Computing*, 14(6):1394–1414, 1993.

[175] Kurt Sebastian and Chi-Wang Shu. Multidomain weno finite difference method with interpolation at subdomain interfaces. *Journal of Scientific Computing*, 19(1-3):405–438, 2003.

[176] L. I. Sedov. Propagation of strong shock waves. *Journal of Applied Mathematics and Mechanics*, 10:241–250, 1946.

[177] L. I. Sedov. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27:1–31, 1978.

[178] Leonid Ivanovich Sedov. *Similarity and dimensional methods in mechanics.* CRC press, 1993.

[179] Matteo Semplice, Armando Coco, and Giovanni Russo. Adaptive mesh refinement for hyperbolic systems based on third-order compact WENO reconstruction. *Journal of Scientific Computing*, 66(2):692–724, 2016.

[180] Varun Shankar, Grady B Wright, Robert M Kirby, and Aaron L Fogelson. A radial basis function (RBF)-finite difference (FD) method for diffusion and reaction–diffusion equations on surfaces. *Journal of Scientific Computing*, 63(3):745–768, 2015.

[181] Chaopeng Shen, Jing-Mei Qiu, and Andrew Christlieb. Adaptive mesh refinement based on high order finite difference weno scheme for multi-scale simulations. *Journal of Computational Physics*, 230(10):3780–3802, 2011.

[182] Chi-Wang Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In Alfio Quarteroni, editor, *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations: Lectures given at the 2nd Session of the Centro Internazionale Matematico Estivo (C.I.M.E.) held in Cetraro, Italy, June 23–28, 1997*, pages 325–432. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[183] Chi-Wang Shu. High order eno and weno schemes for computational fluid dynamics. In *High-order methods for computational physics*, pages 439–582. Springer, 1999.

[184] Chi-Wang Shu. High-order finite difference and finite volume WENO schemes and discontinuous Galerkin methods for CFD. *International Journal of Computational Fluid Dynamics*, 17(2):107–118, 2003.

[185] Chi-Wang Shu. High order weighted essentially nonoscillatory schemes for convection dominated problems. *SIAM review*, 51(1):82–126, 2009.

[186] Chi-Wang Shu. High order WENO and DG methods for time-dependent convection-dominated PDEs: A brief survey of several recent developments. *Journal of Computational Physics*, 316:598–613, 2016.

[187] Chi Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439–471, 1988.

[188] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes, II. *Journal of Computational Physics*, 83(1):32–78, 1989.

[189] Ray Smith. An overview of the tesseract ocr engine. In *Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)*, pages 629–633, 2007.

[190] Gary A Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1–31, 1978.

[191] Thomas Sonar. Optimal recovery using thin plate splines in finite volume methods for the numerical solution of hyperbolic conservation laws. *IMA Journal of Numerical Analysis*, 16(4):549–581, 1996.

[192] Seth C. Huynh Spiegel. A survey of the isentropic Euler vortex problem using high-order methods. Dallas, TX, United States, June 2015.

[193] ASCAC Subcommittee. Top ten exascale research challenges. *US Department Of Energy Report, 2014*, 2014.

[194] A Suresh and HT Huynh. Accurate monotonicity-preserving schemes with runge–kutta time stepping. *Journal of Computational Physics*, 136(1):83–99, 1997.

[195] a. Suresh and H.T. Huynh. Accurate monotonicity-preserving schemes with Runge-Kutta time stepping. *Journal of Computational Physics*, 136(1):83–99, 1997.

[196] Kazuya Takahashi and Shoichi Yamada. Exact Riemann solver for ideal magnetohydrodynamics that can handle all types of intermediate shocks and switch-on/off waves. *Journal of Plasma Physics*, 80(2):255–287, 2014.

[197] Romain Teyssier. Cosmological hydrodynamics with adaptive mesh refinement-a new high resolution code called ramses. *Astronomy & Astrophysics*, 385(1):337–364, 2002.

[198] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, Bee Lim, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[199] Radu Timofte, Rasmus Rothe, and Luc Van Gool. Seven ways to improve example-based single image super resolution. *CoRR*, abs/1511.02228, 2015.

[200] V. Titarev and E. Toro. Finite-volume weno schemes for three-dimensional conservation laws. *Journal of Computational Physics*, 201:238–260, 2004.

[201] V. A. Titarev and E. F. Toro. Finite-volume WENO schemes for three-dimensional conservation laws. *Journal of Computational Physics*, 201(1):238–260, 2004.

[202] V.A. Titarev and E.F. Toro. ADER schemes for three-dimensional nonlinear hyperbolic systems. *Journal of Computational Physics*, 204(2):715 – 736, 2005.

[203] Vladimir A Titarev and Eleuterio F Toro. ADER: Arbitrary high order Godunov approach. *Journal of Scientific Computing*, 17(1):609–618, 2002.

[204] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. pages 839–846, 1998.

[205] EF Toro, RC Millington, and LAM Nejad. Towards very high order Godunov schemes. In *Godunov methods*, pages 907–940. Springer, 2001.

[206] Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction.* Springer Science & Business Media, 2013.

[207] Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction.* Springer Science & Business Media, 2013.

[208] Eleuterio F Toro, M Spruce, and W Speares. Restoration of the contact surface in the HLL-Riemann solver. *Shock Waves*, 4(1):25–34, 1994.

[209] Eleuterio F Toro and Vladimir A Titarev. Derivative Riemann solvers for systems of conservation laws and ADER methods. *Journal of Computational Physics*, 212(1):150–165, 2006.

[210] L. Trefethen and D. Bau. *Numerical linear Algrebra.* Society for Industrial and Applied Mathematics, 1997.

[211] Petros Tzeferacos, Milad Fatenejad, Norbert Flocke, Carlo Graziani, G Gregori, DQ Lamb, D Lee, J Meinecke, A Scopatz, and K Weide. Flash mhd simulations of experiments that study shock-generated magnetic fields. *High Energy Density Physics*, 17:24–31, 2015.

[212] Bart van der Holst and Rony Keppens. Hybrid block-amr in cartesian and curvilinear coordinates: Mhd applications. *Journal of computational physics*, 226(1):925–946, 2007.

[213] Bram Van Leer. Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme. *Journal of Computational Physics*, 14(4):361–370, 1974.

[214] Bram Van Leer. Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection. *Journal of Computational Physics*, 23(3):276–299, 1977.

[215] Bram Van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of Computational Physics*, 32(1):101–136, 1979.

[216] G. Wahba, D. Johnson, F. Gao, and J. Gong. Adaptive tuning of numerical weather prediction models: Randomized GCV in three- and four-dimensional data assimilation. *Monthly Weather Review*, 123:3358–3369, 1995.

[217] Paul Woodward and Phillip Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics*, 54(1):115–173, 1984.

[218] P.r. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics*, 54:115–173, 1984.

[219] Olindo Zanotti, Francesco Fambri, and Michael Dumbser. Solving the relativistic magnetohydrodynamics equations with ADER discontinuous Galerkin methods, a posteriori subcell limiting and adaptive mesh refinement. *Monthly Notices of the Royal Astronomical Society*, 452(3):3010–3029, 2015.

[220] Mengping Zhang and Chi-Wang Shu. An analysis of and a comparison between the discontinuous Galerkin and the spectral finite volume methods. *Computers & Fluids*, 34(4-5):581–592, 2005.

[221] Rui Zhang, Mengping Zhang, and Chi-Wang Shu. On the order of accuracy and numerical performance of two classes of finite volume WENO schemes. *Communications in Computational Physics*, 9(03):807–827, 2011.

[222] Shuhai Zhang, Shufen Jiang, and Chi-Wang Shu. Development of nonlinear weighted compact schemes with increasingly higher order accuracy. *Journal of Computational Physics*, 227(15):7294–7321, 2008.

[223] Tong Zhang and Yu Xi Zheng. Conjecture on the structure of solutions of the Riemann problem for two-dimensional gas dynamics systems. *SIAM Journal on Mathematical Analysis*, 21(3):593–630, 1990.

[224] Weiqun Zhang, Ann Almgren, Vince Beckner, John Bell, Johannes Blaschke, Cy Chan, Marcus Day, Brian Friesen, Kevin Gott, Daniel Graves, Max Katz, Andrew Myers, Tan Nguyen, Andrew Nonaka, Michele Rosso, Samuel Williams, and Michael Zingale. Amrex: a framework for block-structured adaptive mesh refinement. *Journal of Open Source Software*, 4(37):1370, 5 2019.

[225] Weiqun Zhang, Ann Almgren, Marcus Day, Tan Nguyen, John Shalf, and Didem Unat. Boxlib with tiling: an amr software framework. *arXiv preprint arXiv:1604.03570*, 2016.

[226] Jun Zhu and Jianxian Qiu. A new fifth order finite difference WENO scheme for solving hyperbolic conservation laws. *Journal of Computational Physics*, 318:110–121, 2016.

[227] U Ziegler. The nirvana code: Parallel computational mhd with adaptive mesh refinement. *Computer Physics Communications*, 179(4):227–244, 2008.

[228] M Zingale, AS Almgren, MG Barrios Sazo, VE Beckner, JB Bell, B Friesen, AM Jacobs, MP Katz, CM Malone, AJ Nonaka, et al. Meeting the challenges of modeling astrophysical thermonuclear explosions: Castro, maestro, and the amrex astrophysics suite. In *Journal of Physics: Conference Series*, volume 1031, page 012024. IOP Publishing, 2018.