Fast 3D Image Moments

William Diggin        Michael Diggin

December 2020

Abstract

An algorithm to efficiently compute the moments of volumetric images is disclosed. The approach demonstrates a reduction in processing time by reducing the computational complexity significantly. Specifically, the algorithm reduces multiplicative complexity from $O(n^3)$ to $O(n)$. Several 2D projection images of the 3D volume are generated. The algorithm computes a set of 2D moments from those 2D images. Those 2D moments are then used to derive the 3D volumetric moments. Examples of use in MRI or CT and related analysis demonstrates the benefit of the Discrete Projection Moment Algorithm. The approach is also useful in computing the moments of a 3D object using a small set of 2D tomographic images of that object.

1 Introduction

Two dimensional image moments were first defined by Hu [9]. These raw moments are used to derive 7 moment invariants. These invariants are independent of position, scale and orientation. They are useful as feature descriptors for objects in images. For a 2-D image $I(i,j)$ of size $M \times N$, where $0 \leq i < M, 0 \leq j < N$ the discrete moment generating equation of order $(p+q)$ is given by:

$$M_{pq} = \sum_{i,j} I(i,j) i^p j^q \tag{1}$$

Three dimensional moment invariants were proposed by Sadjadi and Hall [16] that extend Hu’s 2D geometric moments to support invariant object recognition for solid homogeneous objects and objects of varying density. For a volume $V(x,y,z)$ of size $L \cdot M \cdot N$, where $0 \leq x < L, 0 \leq y < M, 0 \leq z < N$, the discrete moment generating equation of order $(p+q+r)$ is given by:

$$\mathcal{M}_{pqr} = \sum_{x,y,z} V(x,y,z) x^p y^q z^r \tag{2}$$

Flusser et al [20] used a tensor and graph method to derive 1185 3D moment invariants up to order 16. These moments are invariant to translation, rotation
and scaling. The research demonstrated a volume and surface based approach to their computation.

The medical domain has been an early adopter of 3D image processing, based on tomography and related areas. It is therefore not surprising that much of the research and development of using 3D moments has radiology and anatomy based applications. Reeves et al [14] used 3D densitometric moments to assist with pulmonary nodule measurement (position, orientations, size, etc) and analysis in tomography-based applications. Moments and moment-based statistics were used to measure the changes in related artifacts for the purposes of early diagnosis and intervention.

A method to use geometric moment invariants to assist in segmenting brain structures in MRI sequences provides benefits for medical image analysis and the diagnosis of neurological conditions or diseases [10]. An important research topic is registration of anatomical images, intra- or inter- subject and the use of 3D moments to provide attribute vectors that aid in the correspondence of anatomical structures [18]. Others have used 3D moment invariants to assist with morphometry of brain anatomical entities, e.g. cortical sulci [12], allowing characterization of handedness, etc. In another example, 3D moments up to 6th order were used to generate invariant feature descriptors and were applied to fMRI data to characterise activations in different neural regions of interest [13]. Other researchers have used 3D moments to detect and track vascular networks in MRA [15] or to first segment and then track arteries in CTA [3] - typically using lower order moments for geometric measurements of position, scale and orientation. Outside the medical domain, 3D moments have been used to generate geometric feature vectors that provide indexing into a CAD library or database of parts [4].

In section 2 we review the state of the art for 3D moment computation. In section 3 we outline the Discrete Projection Moment Algorithm to compute the 3D moments efficiently. Then an MRI example is used to showcase the approach. In section 5 some of the practical implementation details are discussed followed by a presentation of comparative results that demonstrate the efficacy of the algorithm relative to the direct naïve approach and with the OpenCV Library [2]. Finally, in section 7 we conduct some analysis of the results and capture some of the learnings.

2 Comparative Computation Methods

Several researchers have proposed novel methods for the fast and efficient computation of moments for volumetric images. These can be considered as surface-based, composition-based or hardware-based. The surface-based approaches generally treat the 3D object as binary and uses variations of Gauss’s Theorem to compute the moments from the boundaries of the object(s). The composition-based approaches operate in regions and while suitable for binary images, could also consider voxel intensities as part of the moment computation. Hardware-based approaches seek to design dedicated processing units and associated al-
algorithms for the purposes of rapid computation. The focus of this review is to consider 1-2 leading examples in each approach.

2.1 Surface-based

Yang et al. [21] [22] developed a method using a existing surface tracking algorithm along with a discrete implementation of Gauss’ Theorem. On binary volumes, they claim that their algorithm reduces the computational complexity from $O(n^3)$ to $O(n^2)$ - although not including the computational cost of the surface tracking method.

2.2 Composition-based

Sossa-Azuela et al. [19] use a morphological operation approach to generate a partition. The partition is a set of cubes that represent regions in the volume. The moments of the volume are simple computations of the moments of the cubes. The researchers claim that the computational complexity is $O(n)$. This is true where the volume is already pre-processed into cubes/regions. However that partitioning step requires a brute-force computation. The cubes/regions are assumed to be either binary or contain the same voxel value, and therefore this algorithm is constrained by those limitations.

Li and Ma [1] presented a method that uses linear transforms on the 3D image. The moments of the linear transforms are computed and the result transformed to conventional moments. While the method demonstrated a reduction in computation, it is limited to 3D binary images.

The OpenCV Library [2] has optimised algorithms for 2D image moment computation. Extending these algorithms to the 3D case not only provides a composition-based solution, but also a non-binary computational approach. The extended algorithm for 3D will be analysed in this paper.

2.3 Hardware-based

Berjón et al. [6] developed a GPU based approach to compute moments of binary volumetric images. Employing pre-computed lookup tables, they reported a 30x improvement over a CPU based approach in many cases. Liu et al. [11] designed a systolic array for the computation of 3D moments. Using a 3-stage system, this is able to compute the raw moments for binary and grey-level images as well as their invariant moments.

3 Discrete Projection Moment (DPM) Algorithm

Given a volumetric image of size $L \cdot M \cdot N$, to compute its moments up to the $(p+q+r)^{th}$ order $k$ using the direct, naïve method in equation 2 requires $O(k^3 \cdot LMN)$ computations (both additions and multiplications).

However, where the volumetric image is projected onto several 2D integral images, the 2D moments of the integrals can be used to compute the 3D moments.
of the original volume. Let $\mathcal{P}$ be a integral projection of the volume function $V(x, y, z)$ onto some plane $(x', y')$

$$\mathcal{P}(x', y', \theta, \phi) = \int_{z'} V(x, y, z) dz'$$

where

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos \phi \cos \theta & \sin \phi \cos \theta & \sin \theta \\ -\sin \phi & \cos \phi & 0 \\ -\cos \phi \sin \theta & -\sin \phi \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

(3)

The transformation represents a rotation by an angle $\phi$ around the z-axis, followed by a rotation by an angle $\theta$ around the $y'$-axis. The plane $(x', y')$ is oriented by $(\theta, \phi)$ and the volume function $V(x, y, z)$ is integrated along the $z'$-axis normal to that plane. The set of these projections over $0 \leq \theta < \pi$ and $0 \leq \phi < \pi$ is the 3D Radon Transform [5].

3.1 Discrete Projections

We are interested in a small subset of projections in the discrete case. To avoid the computation cost of interpolation, we choose the angles $(\theta, \phi)$ so that the integration aligns with the voxels. The result is a discrete 2D image $I(i, j, \theta, \phi)$ that is an integer summation of its 3D discrete counterpart $V(x, y, z)$ along an axis normal to the 2D image plane:

$$I(i, j, \theta, \phi) = \sum_{i=f(x,y,z)}^{f} \sum_{j=g(x,y,z)}^{g} V(x, y, z)$$

(4)

where $f()$ and $g()$ map from 3D volume space to 2D image space.

Given a volume (Figure 1a) of size $L \cdot M \cdot N$ and if we choose $(\theta = 0, \phi = 0)$ then the volume is projected along the z-axis onto the $(x, y)$ plane, resulting in a 2D integral image of size $L \cdot N$ (Figure 1b):

$$I(i, j, 0, 0) = \sum_{i=x}^{i} \sum_{j=y}^{j} V(x, y, z)$$

(5)

Another convenient plane arises from $(\theta = 0, \phi = \frac{\pi}{2})$, where the volume is projected along the x-axis onto the $(y, z)$ plane, giving a 2D integral image of size $M \cdot N$ (Figure 1b):

$$I(i, j, 0, \frac{\pi}{2}) = \sum_{i=y}^{i} \sum_{j=z}^{j} V(x, y, z)$$

(6)

The third axis-aligned plane uses $(\theta = \frac{\pi}{2}, \phi = 0)$ and projects the volume along the y-axis onto the $(z, x)$ plane, resulting in a 2D integral image of size $N \cdot L$ (Figure 1b):

$$I(i, j, \frac{\pi}{2}, 0) = \sum_{i=z}^{i} \sum_{j=x}^{j} V(x, y, z)$$

(7)
We can further construct discrete projections from aligned axonometric perspectives - for example using \((\theta = \frac{\pi}{4}, \phi = 0)\) results in a 2D integral image of size \((L+N-1)\cdot M\) (Figure 1c):

\[
I(i,j, \frac{\pi}{4}, 0) = \sum_{i=x+z, j=y} V(x, y, z) \quad (8)
\]

Similarly, rotating opposite using \((\theta = -\frac{\pi}{4}, \phi = 0)\) results in a 2D integral image of size \((L+N-1)\cdot M\) but over a different range and with negative indexing in the 2D image (Figure 1c):

\[
I(i,j, -\frac{\pi}{4}, 0) = \sum_{i=x-z, j=y} V(x, y, z) \quad (9)
\]

Six such discrete projections can be generated by aligned rotations of \(\pm\frac{\pi}{4}\) around each axis - equivalent to \(\pm\arctan(1)\). Twelve may be generated from \(\pm\arctan(\frac{1}{\sqrt{2}})\) and \(\pm\arctan(2)\), and so on. Similarly, discrete isometric projections result from rotations such as \((\theta = \frac{\pi}{4}, \phi = \arctan(\frac{1}{\sqrt{2}}))\), the zero-padded image size is \((L+N-1)\cdot(L+2M+N-3)\) (Figure 1d):

\[
I(i,j, \frac{\pi}{4}, \arctan(\frac{1}{\sqrt{2}})) = \sum_{i=x+z, j=2y+x-z} V(x, y, z)
\]

### 3.2 Moment Computation

The 2D moments \((M_{\theta, \phi}^{p,q})\) of the projection images can be computed and used to determine the 3D moments of the volumetric image. From equations (1) and (5), the moments of the projection onto the plane \((x, y)\) are:

\[
M_{pq}^{0,0} = \sum_{i,j} I(i,j, 0, 0) i^p j^q
\]

\[
M_{pq}^{0,0} = \sum_{i,j} (\sum_z V(i,j,z)) i^p j^q
\]

\[
M_{pq}^{0,0} = \sum_{i,j,z} V(i,j,z) i^p j^q z^0
\]

\[
\therefore \quad M_{pq0} = M_{pq}^{0,0}
\]

From equations (6) and (7), the 2D moments of projections onto the planes \((y, z)\) and \((z, x)\) are also used to determine some of the 3D moments:

\[
M_{0pq} = M_{pq}^{0,0} \quad (11)
\]

\[
M_{q0p} = M_{pq}^{0,0} \quad (12)
\]
Figure 1: A volume $V(x, y, z)$ and various orthographic projections

The 2D moments of these three projections allow the computation of most of the 3D moments. The exception is for moments where all axes $xyz$ are required. Up to the 4th order these include: $M_{111}$, $M_{211}$, $M_{121}$ and $M_{112}$. The derivation for $M_{111}$ is as follows. Expanding the moment $M_{21}^{1\alpha}$ and using
Equation (8):

\[
M_{\pi 4:0}^{21} = \sum_{i,j} I(i,j, \pi 4) i^2 j
\]

\[
M_{\pi 4:0}^{21} = \sum_{i,j} \left( \sum_{i=x+y} V(x, y, z) \right) i^2 j
\]

\[
M_{\pi 4:0}^{21} = \sum_{x,y,z} V(x, y, z)(x + z)^2 y
\]

\[
M_{\pi 4:0}^{21} = \sum_{x,y,z} V(x, y, z)(x^2 y + 2xyz + yz^2)
\]

\[
M_{\pi 4:0}^{21} = M_{210} + 2M_{111} + M_{012}
\]

\[
\therefore M_{111} = (M_{210} - M_{210} - M_{012})/2
\] (13)

Similarly, by using the 4th order moments \(M_{220}^{\pi 0}, M_{220}^{\pi 0}, M_{310}^{\pi 0}, M_{310}^{\pi 0}, \) etc, of the 2D projections, the following can be derived:

\[
M_{121} = (M_{220}^{\pi 0} - M_{220} - M_{022})/2
\] (14)

\[
M_{112} = (M_{310}^{\pi 0} - M_{310}^{\pi 0} - 2M_{013})/6
\] (15)

\[
M_{211} = (M_{310}^{\pi 0} + M_{310}^{\pi 0} - 2M_{310})/6
\] (16)

Using other discrete and orthographic integral projections along with their 2D moments, formulae for higher order 3D moments can be derived and their values computed efficiently. Furthermore, the computation of the 2D moments of the projections can be accomplished efficiently using a Discrete Radon Transform based algorithm [7].

Given several 2D projections of the 3D volume will be used to compute the 3D moments, it is possible to invert the normal construction. Instead of performing a full tomographic sweep of the 3D artifact, several chosen 2D tomographic projections can be used instead of 3D reconstruction. This set of projections could be used to determine the 3D moments [17] and therefore reduce the initial data acquisition demands and associated complexity.

4 Example

To visualise an example of the DPM algorithm in practice, a volumetric image of a brain MRI was used, Figure 2. The object has been imaged with a 1mm spacing in the coronal and saggital directions and with a 0.4mm spacing in the axial direction. The image dimensions are 512x512x230 voxels.

To compute the 4th order 3D moments, we require five projections. We choose, for economies of computation, projections based on equations (5), (6), (7), (8) and (9). This leads to five 2D images as presented in Figures 3, 4 and 5. These 2D images are generated by indexing through the volume data and
accumulating the voxel values into each image. Figures 3b, 3c and 3d provide additional visualisation perspectives to the 2D projection onto the (x,y) plane. The accumulation is very efficient as it is performed with pure addition.

From these five 2D images, we now wish to compute their 2D moments. For this, we choose the efficient DRT algorithm [7] that further projects each of the 2D images to a series of 1D integrals. Again, this step is an efficient process as it too is performed with pure addition. Based on that DRT algorithm, the moments of these 1D integrals are computed and then combined to generate all the 4th order 2D moments for these five 2D images.

Finally, using equations (10), (11), (12), (13), (15), (14) and (16), we can compute all the 3rd or 4th order 3D moments of the Brain MRI. Total computation time is $\approx 20\text{ms}$ for 3rd order and is $\approx 30\text{ms}$ for the 4th order.

Computation of the object features, such as centroid, is now possible:

$$\bar{x} = \frac{M_{100}}{M_{000}} \quad \bar{y} = \frac{M_{010}}{M_{000}} \quad \bar{z} = \frac{M_{001}}{M_{000}}$$

From these, the central moments, $\mu_{pqr}$, may be computed:

$$\mu_{pqr} = \sum V(x, y, z)(x - \bar{x})^p(y - \bar{y})^q(z - \bar{z})^r$$

The central moments are invariant to translation. The central moments can then be normalised for scale invariance by:

$$\eta_{pqr} = \frac{\mu_{pqr}}{\mu_{000}^{\frac{p+q+r}{3}}} + 1 \quad \text{(17)}$$
However, in most tomography-based imaging systems, the physical spacing is programmable. In an MRI DICOM sequence, axial imaging is constituted with a series of, say, 512x512 images. In this case, each image element may have a physical spacing of 1.0mm in the $x-, y$-axes and each slice in the axial plane (or $z$-axis) may have a physical spacing of 0.4mm. This is representative of our example. This means that the computed moments from the DPM algorithm are only valid for the virtual voxel spacing. To compensate for the scale mapping from virtual to physical, we can normalise the computed central moments as follows:

$$
\mu'_{pqr} = \alpha^{1+p}\beta^{1+q}\delta^{1+r}\mu_{pqr}
$$

The quantities, $\alpha$, $\beta$ and $\delta$ are the scale factors in each image axis.

Given the moments are now available to us relative to the physical coordinate frame, we can compute several other metrics, such as orientation, elongation (length/width ratios), sphere of gyration, as well as feature descriptors that are invariant to translation, rotation and scale [16], [20].

An interesting avenue for further investigation revolves around performing pre-processing in projection or integral space. For example, rather than conducting background segmentation or subtraction on the 3D volume, we can...
perform that operation on the projection images or the 1D integrals. Obviously there are resulting savings in computational complexity and processing time.

5 Implementation Details

To take advantage of the computational efficiencies of the DPM algorithm, we outline some of the important implementation details to assist in construction. The pseudo-code to generate the projections is given in Listing 1. This shows the mechanism by which the 2D projection images are constructed. It is clear that pure discrete addition of voxels to pixels occurs.

```
// Volume = 3D image volume
// XY, YZ, ZX, X_plusZY
// and X_minus_ZY = 2D projection images
For k = 1 to Depth
  For j = 1 to Height
    For i = 1 to Width
      XY[i, j] += Volume[i, j, k]
```

```
The DRT algorithm to compute the 2D moments projects each of the 2D images computed in Listing 1 onto five 1D integrals as illustrated in Listing 2. The accumulation into the integrals is a pure discrete additive process. Therefore, the first two steps in the computation chain are straightforward and trivial to construct.

By iterating linearly through the 3D volume data, the 2D image data and the 1D integral data, the algorithm capitalises on processor caching and pipelining optimisation and so captures significant time advantages.

For 4th order, the moments of each of the five 2D images are computed according to the DRT algorithm [8]. For each image, the five 1D integrals are used to compute their equivalent 1D moments. These are then combined to provide the 2D moments for each image.

The 2D moments, $M_{pq}^{x0}$, from the projection in Equation (9) need to be adjusted due to the requirement of negative indexing. This simply means an offset so that the moment origin is positioned at $\hat{x} = N - 1$ in the 2D image.
This can be accomplished for the relevant moments as follows:

\[
M_{31}^{\frac{-\pi}{4}:0} = M_{31}^{\frac{-\pi}{4}:0} - 3M_{21}^{\frac{-\pi}{4}:0}\hat{x} + 3M_{11}^{\frac{-\pi}{4}:0}\hat{x}^2 - M_{01}^{\frac{-\pi}{4}:0}\hat{x}^3
\]

\[
M_{13}^{\frac{-\pi}{4}:0} = M_{13}^{\frac{-\pi}{4}:0} - M_{03}^{\frac{-\pi}{4}:0}\hat{x}
\]

Finally, use equations (10), (11), (12), (13), (15), (14) and (16) to compute all the relevant 3D moments.

## 6 Results

We implemented the naïve computational approach, based on Equation (2), to provide a reference for computational cost. In addition, as the OpenCV Library is the only other available source of reference to efficiently compute grayscale moments, we extended its 2D OpenCV moment implementation. The base algorithm used in the OpenCV library was extended to compute the 3rd and 4th order 3D moments. The naïve and OpenCV algorithms were measured and compared against the DPM algorithm detailed in this paper. All the algorithms have been coded, compiled and optimised for the test processor instruction set. Then, using a large sample size to maximise processor availability, we selected the fastest exemplar for each algorithm and each test candidate. The comparison of execution times was gathered for differing volumetric image sizes, measured in MegaBytes (MB). All algorithms were executed on an Intel Core i5 7th Generation CPU.

| Size (MB) | Naïve (ms) | OpenCV (ms) | DPM (ms) |
|-----------|------------|-------------|----------|
|           | 3rd        | 4th         | 3rd      | 4th      |
| 1         | 11         | 23          | 1.7      | 2.0      | 0.3 | 0.4 |
| 2         | 23         | 46          | 3.4      | 4.0      | 0.6 | 0.9 |
| 4         | 47         | 94          | 6.7      | 8.1      | 1.0 | 1.5 |
| 8         | 95         | 188         | 13.6     | 16.8     | 2.0 | 2.8 |
| 16        | 192        | 377         | 28.0     | 34.1     | 4.0 | 5.3 |
| 32        | 386        | 760         | 57.0     | 69.0     | 8.3 | 11.0 |
| 64        | 772        | 1447        | 115.9    | 130.4    | 17.6 | 21.2 |
| 128       | 1466       | 2885        | 221.5    | 262.0    | 37.5 | 41.0 |
| 256       | 2927       | 5793        | 445.0    | 525.2    | 63.5 | 84.6 |
| 512       | 5905       | 12200       | 892.1    | 1052.4   | 130.4 | 173.8 |
| 1024      | 12772      | 24646       | 1933.0   | 2198.0   | 273.5 | 400.1 |

Table 1: Comparative Computational Times vs Volumetric Image Sizes for Naïve, OpenCV and DPM for 3rd and 4th order 3D moments.

Table 1 and Figure 6 show the measurements and plot comparing the time needed to compute the 3D moments of the 3rd and 4th order for a given volume size. The smallest volume used was a 128x128x64 voxel image, with the
Figure 6: Comparative Computational Cost Vs Image Sizes for Naïve, OpenCV and DPM.

largest at 1024x1024x1024 voxels. The y-axis illustrates the execution time in milliseconds. The secondary y-axis is also in milliseconds but in $\log_{10}$ for easier plotting. The x-axis is the volume data size in MegaBytes.

It is evident that the naïve approach does not scale well. The cost to execute the $4^{th}$ order moments using the naïve approach is twice the cost for the $3^{rd}$ order. It requires 12 and 24 seconds to compute the $3^{rd}$ and $4^{th}$ order moments for a 1GB volume.

The OpenCV algorithm performs significantly better, circa 6x to 10x better than the naïve approach. Is has only a $\approx 20\%$ overhead when computing the $4^{th}$ order versus the $3^{rd}$ order. It requires $\approx 1.9$ and $\approx 2.2$ seconds to compute the $3^{rd}$ and $4^{th}$ order moments for a 1GB volume.

The DPM algorithm performs best, 5x to 10x better than the OpenCV approach. It has a $\approx 30\%$ overhead when computing the $4^{th}$ order versus the $3^{rd}$ order. The DPM algorithm needed only $\approx 0.3$ and 0.4 seconds to compute the $3^{rd}$ and $4^{th}$ order moments for a 1GB volume.
Note that all algorithms exhibit an approximate doubling of processing time for an equivalent doubling in data size.

7 Analysis

To simplify the analysis, consider an $n \times n \times n$ volume. The naïve implementation to compute all 3D moments up to the 4th order using equation (2) requires $37n^3$ multiplication and $35n^3$ additions. Extending the OpenCV implementation from 2D to 3D, the same moment computation requires $4n^3 + 14n^2 + 20n$ multiplications with $5n^3 + 20n^2 + 20n$ additions.

As summarised in Table 2 for the DPM algorithm, the generation of the 2D images from the 3D volume is computationally trivial - requiring $4n^3$ additions for 3rd order and $5n^3$ additions for 4th order. No multiplications are required for this step.

For the second step, the construction of the 1D integrals of the 2D images is also computationally trivial. For all 2D images, it requires $28n^2$ additions for 3rd order and $35n^2$ additions for 4th order. Again, no multiplications are involved.

The computation of the moments from the 1D integrals requires $54n$ multiplications for 3rd order and $126n$ multiplications for 4th order. The computation needs $60n$ additions for 3rd order and $132n$ additions for 4th order. Following these steps is the computation of the 2D moments from the 1D integral moments and then the 3D moments from the 2D moments. This is a constant computation and is straightforward.

| Method | $+/\times$ | 3rd Order | 4th Order |
|--------|------------|-----------|-----------|
| Naïve  | $+$        | $20n^3$   | $35n^4$   |
| Naïve  | $\times$   | $21n^3$   | $37n^4$   |
| OpenCV | $+$        | $4n^3 + 10n^2 + 20n$ | $5n^3 + 20n^2 + 20n$ |
| OpenCV | $\times$   | $3n^3 + 7n^2 + 11n$ | $4n^3 + 14n^2 + 20n$ |
| DPM    | $+$        | $4n^3 + 28n^2 + 60n$ | $5n^3 + 35n^2 + 132n$ |
| DPM    | $\times$   | $54n$     | $126n$    |

Table 2: Scale of multiplications and additions of the Naïve, OpenCV and DPM for 3rd and 4th order 3D moments.

Therefore for a volume of size $400 \times 400 \times 400$, naïve vs OpenCV vs DPM have the following multiplication ratios for 3rd order moments approximately: $62,222 : 8,888 : 1$. The multiplication ratios for 4th order moments approximately are $188,000 : 5,000 : 1$. The computation for 3rd and 4th order addition is $4n^3$ and $5n^3$ for the OpenCV method and the DPM algorithm - although the naïve method requires $\approx 5x$ and $\approx 7x$ relative to the others.

The naïve and OpenCV approaches both exhibit $O(n^3)$ multiplicative complexity whereas the DPM algorithm is $O(n)$. All methods are $O(n^3)$ additive.
complexity. In practical terms, each algorithm has $O(n^3)$ complexity to load data for processing and store the results after computation. The DPM algorithm reduces the total number of operations by eliminating much of the per voxel multiplication. This is the main reason for its superior performance.

8 Conclusion

The DPM algorithm provides significant computational advantages as volume size increases. It is shown to be $50-60\times$ faster than the direct naïve computation and $5-7\times$ faster than an optimal OpenCV algorithm. The DPM algorithm performs well for both binary or grey-level volumetric images and does not require any surface tracking or morphological pre-processing. The approach demonstrates an efficient means to generate $3^{rd}$ and $4^{th}$ order moments of 3D artifacts. By generating other discrete projections we can extend the algorithm to higher order moments, while retaining computational efficiency. The resultant moment invariants have uses as feature descriptors, ideally as input to classifiers or neural inference engines.

Further work to extend the approach to higher order moments in a formal and general fashion is warranted. So too is extending the algorithmic approach to multi-dimensional data-sets, e.g. temporal related signals. More simply, there are opportunities to capitalise on other transformations to the data in projection or integral space and then compute the resulting moments. This is instead of conducting those transformations in higher-dimensional space and suffering the resulting computational cost.

9 Acknowledgements

The authors wish to acknowledge the volume datasets provided by the Naval Postgraduate School at [https://savage.nps.edu/Savage/VolumeRendering/datasets/nrrd/](https://savage.nps.edu/Savage/VolumeRendering/datasets/nrrd/). The authors also acknowledge the datasets provided via [https://slicer.org](https://slicer.org) - contributors: Steve Pieper (Isomics), Benjamin Long (Kitware) and Jean-Christophe Fillion-Robin (Kitware).

References

[1] BingCheng Li and Song De Ma. “Efficient computation of 3D moments”. In: Proceedings of 12th International Conference on Pattern Recognition. Vol. 1. 1994, 22–26 vol.1. doi: [10.1109/ICPR.1994.576218](https://doi.org/10.1109/ICPR.1994.576218).

[2] G Bradski. “The OpenCV Library”. In: Dr. Dobb’s Journal of Software Tools (2000).
[3] Kun Chen et al. “Coronary Artery Segmentation Using Geometric Moments based Tracking and Snake-Driven Refinement”. In: Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference 2010 (Aug. 2010), pp. 3133–7. DOI: 10.1109/IEMBS.2010.5627192

[4] George Cybenko, Aditya Bhasin, and Kurt Cohen. “Pattern Recognition of 3D CAD Objects : Towards an Electronic Yellow Pages of Mechanical Parts”. In: Smart Engineering Systems Design 1 (Sept. 1999).

[5] S.R. Deans. The Radon Transform and Some of Its Applications. Dover Books on Mathematics Series. Dover Publications, Incorporated, 2013. ISBN: 9780486788548. URL: https://books.google.ie/books?id=BXiXswEACAAJ

[6] Daniel Berjón Díez, Sergio Arnaldo Duart, and Francisco Morán Burgos. “A parallel implementation of 3D Zernike moment analysis”. In: Proceedings of the SPIE. Parallel Processing for Imaging Applications. Vol. 7872. SPIE, 2011. URL: http://oa.upm.es/12233/

[7] W Diggin and M Diggin. Using the discrete radon transformation for grayscale image moments. 2020. arXiv: 2008.11083 [cs.CV].

[8] William Diggin and Michael Diggin. Efficient Computation of Higher Order 2D Image Moments using the Discrete Radon Transform. 2020. arXiv: 2009.09898 [cs.CV].

[9] MK Hu. “Visual pattern recognition by moment invariants”. In: IRE Transactions on Information Theory 8.2 (1962), pp. 179–187.

[10] Mostafa Jabarouti Moghaddam and Hamid Soltanian-Zadeh. “Automatic Segmentation of Brain Structures Using Geometric Moment Invariants and Artificial Neural Networks”. In: vol. 21. Feb. 2009, pp. 326–37. ISBN: 978-3-642-02497-9. DOI: 10.1007/978-3-642-02498-6_27.

[11] Jianguo Liu, Shaobo Chen, and Hai Wang. “Global systolic array for fast computation of 3D moment invariants”. In: Proc SPIE (Oct. 2009). DOI: 10.1117/12.833710

[12] Jean-François Mangin et al. “Brain morphometry using 3D moment invariants”. In: Medical Image Analysis 8 (Sept. 2004), pp. 187–196. DOI: 10.1016/j.media.2004.06.016.

[13] Bernard Ng et al. “Characterizing fMRI Activations within Regions of Interest (ROIs) Using 3D Moment Invariants”. In: vol. 2006. July 2006, pp. 63–63. ISBN: 0-7695-2646-2. DOI: 10.1109/CVPRW.2006.52.

[14] Anthony P. Reeves et al. System and method for three-dimensional image rendering and analysis. Sept. 2007. URL: https://www.freepatentsonline.com/7274810.html
[15] Patrick Reuzé et al. “A 3-D moment based approach for blood vessel detection and quantification in MRA”. In: *Technology and health care: official journal of the European Society for Engineering and Medicine* 1 (Nov. 1993). doi: [10.3233/THC-1993-1209](https://doi.org/10.3233/THC-1993-1209).

[16] Firooz Sadjadi and Ernest Hall. “Three-Dimensional Moment Invariants”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 2 (Apr. 1980), pp. 127–136. doi: [10.1109/TPAMI.1980.4766990](https://doi.org/10.1109/TPAMI.1980.4766990).

[17] David B Salzman. “A method of general moments for orienting 2D projections of unknown 3D objects”. In: *Computer Vision, Graphics, and Image Processing* 50.2 (1990), pp. 129–156. issn: [0734-189X](https://www.sciencedirect.com/science/article/pii/0734189X9090038W). doi: [https://doi.org/10.1016/0734-189X(90)90038-W](https://doi.org/10.1016/0734-189X(90)90038-W).

[18] Dinggang Shen and Christos Davatzikos. “HAMMER: Hierarchical attribute matching mechanism for elastic registration”. In: *IEEE transactions on medical imaging* 21 (Dec. 2002), pp. 1421–39. doi: [10.1109/TMI.2002.803111](https://doi.org/10.1109/TMI.2002.803111).

[19] Humberto Sossa-Azuel et al. “3-D Cartesian Geometric Moment Computation using Morphological Operations and its Application to Object Classification”. In: *Ingeniería, investigación y tecnología* 8 (June 2007), pp. 111–123. doi: [10.22201/fi.25940732e.2007.08n2.010](https://doi.org/10.22201/fi.25940732e.2007.08n2.010).

[20] Tomás Suk and Jan Flusser. “Tensor Method for Constructing 3D Moment Invariants”. In: Aug. 2011, pp. 212–219. doi: [10.1007/978-3-642-23678-5_24](https://doi.org/10.1007/978-3-642-23678-5_24).

[21] Luren Yang, Fritz Albregtsen, and Torfinn Taxt. “Fast Computation of 3-D Geometric Moments Using a Discrete Gauss’ Theorem.” In: Jan. 1995, pp. 649–654.

[22] Luren Yang, Fritz Albregtsen, and Torfinn Taxt. “Fast Computation of Three-Dimensional Geometric Moments Using a Discrete Divergence Theorem and a Generalization to Higher Dimensions”. In: *CVGIP: Graphical Model and Image Processing* 59 (Mar. 1997), pp. 97–108. doi: [10.1006/gmip.1997.0418](https://doi.org/10.1006/gmip.1997.0418).