PRAMANA — journal of physics © Indian Academy of Sciences Vol. 85, No. 6 December 2015 pp. 1157–1180

# Physics-based vicinity emulation for enhanced perception in teleoperation

J K MUKHERJEE
Electronics and Instrumentation Services Division, Bhabha Atomic Research Centre, Mumbai 400 085, India
E-mail: jkmukh@yahoo.co.uk

MS received 2 August 2013; revised 21 January 2014; accepted 27 September 2014
**DOI:** 10.1007/s12043-015-0957-0; ***e*Publication:** 25 July 2015

**Abstract.** A workspace modelling technique, that achieves real-time detection of closeness of robot to workspace object, has been developed by additional physics attribute attachment to workspace. Physics phenomena suiting the realistic work conditions have been identified and a computational model for emulating the phenomenon has been developed. Need for building sensitivity to dynamics in the model has also been addressed. Model building and algorithms for embedding the model in workspace comprising complex object shapes have been studied in detail. Applications like fast execution in real-time scenarios, design of data structure for realistic implementation and the problem of attaining functionality without new sensor addition to telerobots working in radioactive environment are addressed.

**Keywords.** Perception; viscosity; physics model; computation; telerobotics; master–slave manipulator.

**PACS Nos 07.05.Tp; 07.07.Tw**

## 1. Introduction

In coupled master–slave type robotic systems that are used in telerobotic working domain with master being actuated by human operator and slave arm following it, force feedback approaches have been used in several ways [1,2]. The following error- as well as slave velocity-based impedance modification of two port control networks have been used [3,4].

Force felt by the master arm operator (figure 1) is given as

$$\mathbf{M}x'' + \mathbf{B}x' + \left(\frac{1}{\mathbf{A}}\right)\mathbf{F}_{\mathrm{e}} = \mathbf{F}_{\mathrm{h}}. \tag{1}$$

The first two terms denote the mass and damping matrices, respectively for the master arm. The two external stimuli to the master robot include the force applied by human and interaction force between the slave robot and its environment. The scale **A** is the force amplification between the master and the slave robots. The role of the master robot is to

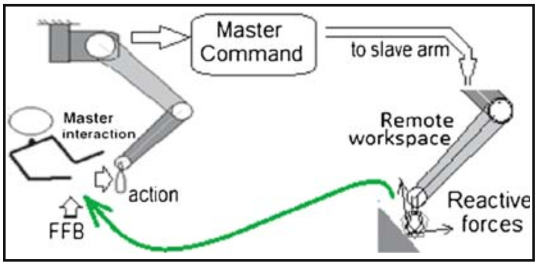


**Figure 1.** Robot arm teleoperated by the master.

provide position commands to the slave manipulator as well as to provide force feedback from the remote environment to the operator [5,6].

Telerobotic front-ends used, in nuclear application domain, face radiation damage [7,8]. These applications cannot use CCD imager-based close range viewing [9].

Contact-type sensing provided by the force torque sensors in the wrist are used but do not serve for detecting nearness to an object and produce effects that are only reactive in nature. The third term in (1) is an attenuated version of these effects. For balanced arm, when it is not in contact with any workspace part or object but very close to the part, the third term is zero. Our aim is to add a term in (1) that is sensitive to the robot's relative position *vis-à-vis* a part in the workspace as well as robot's dynamics. This term must be active in pre-contact phase and its magnitude must be based on the proximity to the object and speed of approach. It can help the operator to perceive the closeness of the slave arm to object bodies in remote workspace. The method must use only the existing sensors in a telerobot slave and avoid adding any more. The technique aims at building workspace models with additional attributes for facilitating this objective in real-time operation regimes.

## 2. Previous work

The parts in a slave robot's workspace can be of two types: (1) needing manipulation and (2) not allowing any manipulation. For parts of the latter type, barrier or 'virtual fixture' that work in software form as added constraint has been used. 'Virtual fixture' refers to a general class of guidance modes [10,11] that help a robotic manipulator to perform a task by limiting its movement in restricted regions and/or influencing its movement along the desired paths and prohibit the motion of a robot manipulator in forbidden regions of geometric or configuration space.

Earlier work [12] used virtual barrier objects in workspace to restrict the operator from moving the master arm in selected areas by constraining master motion. Distance to virtual barrier was computed online and was used as a measure of developing opposition to master's motion. The barrier method has been used with limited number of plains or hyperplains in case of constraints in joint space, to achieve real-time performance by minimizing fixtures and limiting them to simple types.

## 3. Physics inspired model

At first glance, the artificial force phenomena that are akin to the repulsion caused by virtual barrier, appears attractive as a physics-based approach to form sensorless tactile

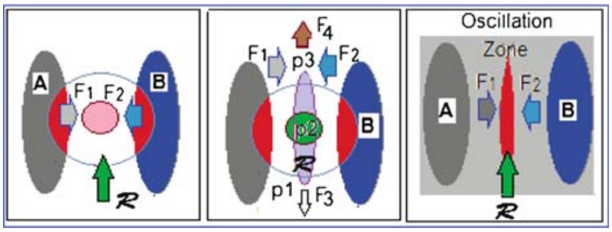


**Figure 2.** Robot moving through a force field.

reaction to operator. Potential field-based approaches also use some form of force created by interactions of a charged mobile robot with the field and are popular for autonomous navigation of mobile robots [13]. But these have drawbacks [14]. Simulated objects with spring action on the surface also have been used earlier for force feedback [15,16] formation. In these approaches, broadly speaking, some form of force field (figure 2) exists. The methods are active in nature implying that at any location within the field, forces will act on a point robot irrespective of its velocity. The temporal variation in force is purely an attribute of spatial state variation of $\mathscr{R}$. The method is sensitive to robot position *vis-à-vis* the repelling object. It continues to push the master arm back out of the repulsion zone even if $\mathscr{R}$ stops and attempts to maintain position in this zone. Force feedback accruing from closely placed objects or concave zones in objects cause instability of control by subjecting the master arm to mutually counteracting forces. Force field is directed in nature and creates undesirable effects. For instance, $\mathscr{R}$ faces a push and pull effect in the periphery of force field at $p_1$ and $p_3$, respectively, leading to oscillatory reaction (figure 2).

### 3.1 *A new approach*

An alternative drag-based approach is investigated and a prototype has been developed. A body with surface area $S$ in contact with a fluid of viscosity $\mu$, if moves at velocity $\mathbf{V}$ faces an opposing drag force $\mathbf{F}$ (figure 3).

$$\mathbf{F} = \mu * S * \mathbf{V}. \tag{2}$$

Visualize a case with object $\mathscr{O}$ surrounded by fluids of different viscosities (figure 4). Point robot $\mathscr{R}$ faces viscous drag that varies as $\mathscr{O}$ is approached [17]. $\mathscr{R}_1$ and $\mathscr{R}_2$ encounter the same opposing forces if their velocities are the same (assuming they have

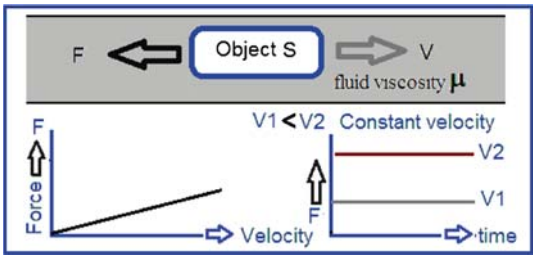


**Figure 3.** Object faces drag while moving in a viscous medium.

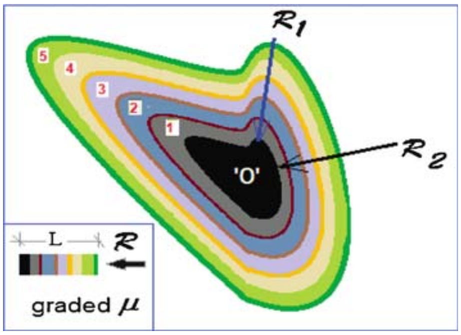


**Figure 4.** Object 'O' surrounded by different viscosity fluids.

the same surface $S$). The viscosities in figure 4 have five discrete values but it can change continuously as a function of $d$

$$\mu = f\,(d)\,. \tag{3}$$

Here $d$ is the distance from the surface of the nearest object.
Force $\mathbf{F}$ acting on the point robot $\mathscr{R}$ located at position $P(x, y, z)$ where the viscosity is $\mu_{\rm p}$, the drag on it is given as

$$\mathbf{F}_{\rm p} = \mu_{\rm p} * S * \mathbf{V}, \tag{4}$$

where $S$ is the surface area of $R$, $\mathbf{V}$ is the velocity and direction of $\mathbf{F}$ is opposite to $\mathbf{V}$,

$$\mathbf{F}_{\rm p} = -1 * f\,(d) * S * \mathbf{V}. \tag{5}$$

A discrete form of $f(d)$ is considered for $q$ (integer) number of discrete viscosity layers (see figure 4). The viscosities $\mu_n$ are chosen with $\mu_1$ of high magnitude for the first layer and for subsequent layers a graded viscosity such that $\mu_n < \mu_{(n-1)}$ for $n$ ranging from 2 to $q$. Any $\mathscr{R}$ moving at constant speed faces stepped increase in drag over the layers (figure 5).

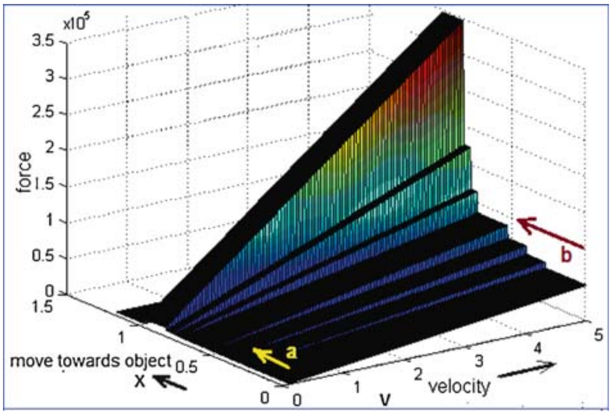


**Figure 5.** Drag force behaviour as a function of speed of approach and degree of closeness to the object.

3.1.1 *Model behaviour.* A slow moving $\mathscr{R}$ following the path along $a$ encounters very low drag. The drag effect is more enhanced for $\mathscr{R}$ moving at higher velocity as it moves along path b. This model retains the property of velocity dependence of the drag. The position-dependent change in $\mathbf{F}$ can be used to generate perceivable vicinity effect with increasing nearness to a part while approaching it and the property can be used to present a tactile effect on the operator's hand.

The $\mathscr{R}$ moving towards an object with high velocity under the master action faces high drag force but on stopping, the operating point shifts from $b$ to $a$ and the drag reduces. Contrary to the force field approach, here $\mathscr{R}$ is not pushed back by the environment and so no force feedback acts on the master arm. The elimination of 'conflict' with workspace environment at zero speed gives stability to the master side as well as the complete system unlike the force field-based environment that introduces instability in dense object neighbourhood. It also implies, that an object can be reached at very slow speed by the operator unlike that in the case of force feedback formed in the potential field. This is desirable as very low speed approach does not cause any impact on contact. After contact, the opposition faced by the slave side servo actuators cause $\mathbf{F}_e$ to build up and generate force feedback (1) depending on the rigidity of the object.

## 3.2 *Challenge in forming graded $\mu$ effect*

The distance of $\mathscr{R}$ moving in workspace has to be determined at each instantaneous position with respect to the objects nearby. The corresponding $\mu$ at the location is found and (5) is applied. This is a difficult task in real-time applications. Consider a workspace with objects A, B and C and a point robot $\mathscr{R}$ (figure 6). Surface-based occupancy model is constructed from CAD models or from range-sensed data [12]. These comprise triangular surface segments typically in STL version [18]. $\mathscr{R}$ must be made sensitive to the vicinity of objects in some way. As indicated by (3), direct use of physics principles need that physical distance from the nearest surface be computed when $\mathscr{R}$ is in the vicinity of an object. Vicinity is defined as a zone engulfing the surface on its outer face up to a distance $\delta$ from the surface (green zones).

## 3.3 *Problem in finding applicable $\mu$*

Fast executing procedures have received attention [19]. Reduction by excluding objects (and their component surfaces) that are far from $\mathscr{R}$, has been achieved using a simple

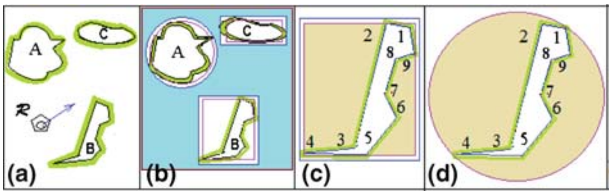


**Figure 6.** General enclosing volume-based neighbour detection schemes and their limited effectiveness.

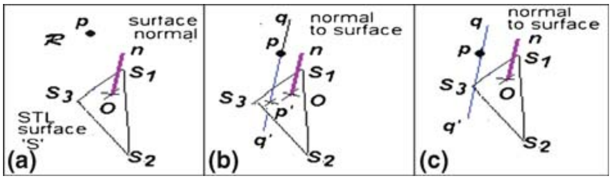


**Figure 7.** Surface distance computation from a point.

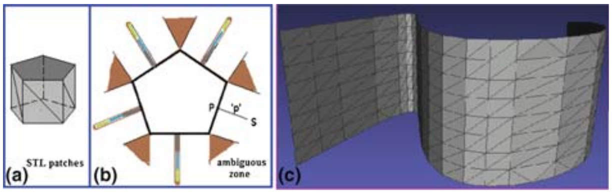


**Figure 8.** (**a**) Pentagonal cylinder in STL format, (**b**) zones of uncertain solution for the nearest surface (in top view of object), (**c**) STL surface for a curved surface for which numerous zones of uncertain solution exist.

test for the presence of $\mathscr{R}$ within the smallest box (aligned with Cartesian workspace coordinate) formed to fully include vicinity extent of the object (figure 6b). Oversized sphere of radius $r$ too, may be used by computing the single-point distance of $\mathscr{R}$ to the centre of the bounding sphere (figure 6c) and comparing $d$ with $r$. Representation of spherical object has received attention earlier too [20,21]. But, near an object, when a moving $\mathscr{R}$ has short time to react, it has to rely on cumbersome computations as large number of surfaces qualify through the above tests.

The computation to find nearest surface [22] is a sequential process. It requires each triangular surface (figure 7) to be considered for computing its unit normal, computing a vector parallel to the normal from the present position of $\mathscr{R}$, computing its intersection with the surface, finding length of this vector and comparing these lengths for all surface triangles to find the nearest of all. The entire process needs to be repeated by changing the location of $\mathscr{R}$. It also faces degeneration problem when $\mathscr{R}$ is in shaded corners leading to the failure in finding the nearest surface. With physical parts in workspace modelled by numerous STL [23] surface triangles (figure 8c) on the object body, the degeneration problem may be severe.

## 4. An efficient approach for ranked vicinity formation

Another spatial distribution-centric view at figure 4 suggests that the space surrounding a workspace object may be imparted property based on a ranked neighbourhood criterion in which the space touching the object surface from outside is ranked 1 and the space touching the rank-1 surface from still outer side is ranked 2. Such ranking can be imparted for layers till an integer number $q$. More is the object prohibited to touch, higher is the value of $q$. The $\mu$ values can then be assigned depending on rank. The vicinity formation is attempted in two steps in this work. The entire Cartesian workspace of $\mathscr{R}$ with the

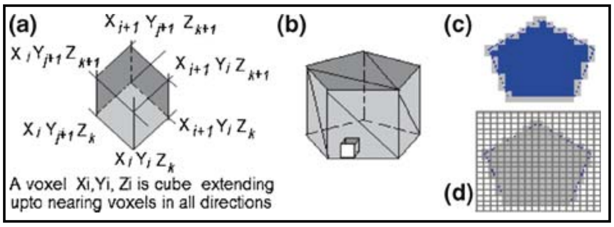


**Figure 9.** (**a**) Voxel definition, (**b**) surface voxel of a cylinder object, (**c**) top layer surface voxel and (**d**) intermediate layer in a cylinder with the body voxel.

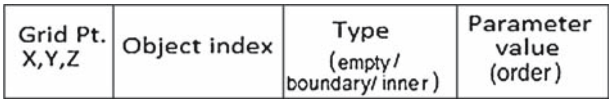
objects appearing at their respective positions is represented by equal-sized cubic volume elements in the first phase and then the space around a workspace object is imparted by varying the $\mu$ property in phase 2.

## 4.1 *Voxel-based modelling*

Voxel-oriented approaches have delivered very encouraging results in 3D graphics [24,25]. In such approaches, workspace is represented as a three-dimensional array of equal-sized cube-shaped volume elements [26] with integer coordinate locations ($X$, $Y$, $Z$) that are referred to as 'Voxel' and denoted as $V_{(XYZ)}$ in further descriptions. A voxel $V_{(XYZ)}$ represents a cube with diagonal nodes $X_i$, $Y_j$, $Z_k$ and $X_{i+1}$, $Y_{i+1}$, $Z_{i+1}$ as shown in figure 9a. The indices range from 0 to an integer needed for representing the full extent of the workspace with desired granularity. The voxels were assigned 'potential' properties for path finding problems in [27]. In this study, we use them to assign other properties. For the representation of spatial occupancy of space by parts in the workspace, we adhere to the following interpretation. A grid position in workspace where no object exists is represented by a free voxel. A voxel through which a triangular surface passes is the boundary voxel 'BV' (figure 9b), a voxel inside an object is the occupied voxel. Typical voxel data are recorded (figure 10) with the last field as a signed integer.

### 4.1.1 *Constructing voxel-based model.*

For CAD models in STL version, data representing the 3D shape of an object are associated with the Cartesian workspace with respect to its pose and position in workspace using node set rotations and translations. This also eliminates the hole-forming problem that can occur if a voxelized model is rotated [28]. The entire workspace voxel array is initialized as 'empty' type. In phase 1, triangular surface segments in CADSTL model or mesh-coded surface triplets from range-based volume occupancy models [12] need to be computationally intersected by grid lines of Cartesian coordinate system over the entire workspace range.



| Grid Pt. X,Y,Z | Object index | Type (empty/ boundary/ inner) | Parameter value (order) |
|---|---|---|---|

**Figure 10.** Data associated with a voxel in a 3D voxel array.

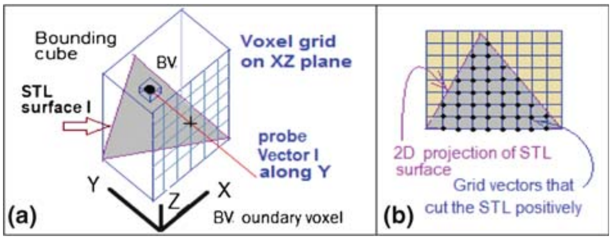


**Figure 11.** Minimizing probe vectors for optimized intersection.

4.1.2 *Optimization in voxel conversion.* Optimization is carried out to improve data quality. Intersection of the triangular plane (STL element) with grid vector is more precise if the intersecting vector is not parallel or near parallel to the surface. Generally, grid vectors of any orientation, i.e., either $X$, $Y$ or $Z$ can be used to intersect the STL triangle [29] but to ensure intersection by grid vector of appropriate orientation, minimum bounding brick is formed with three nodes and the largest rectangle face of the brick is found. Grid line normal to this rectangle and crossing it are used (figures 11a and 11b) to intersect the triangle for $BV$ determination [30,31]. Reliable intersection tests are necessary [32] for accuracy.

4.1.3 *Voxel labelling.* In the second phase, the modified voxel array is analysed completely to label the inner voxels. Voxels at constant $Z$ (integer) levels in $Z_{\max} > Z > Z_{\min}$ zone have closed contour formed by boundary voxels of the object (figure 9c). The voxel data for a $Z$-value are treated like a 2D image as they form a planar slice of the object. These have closed contours formed by the earlier marked $BV$. By using raster scan method [25] the inner voxel is marked as $IV$. The operation over $Z_{\min}$–$Z_{\max}$ results in full voxelization of part in 3D form. Object serial number is also appended as 'object index' in labelled voxels. Gray voxels $BV$ in figure 9c arise from the side and those in figure 9d from top surfaces of part in figure 9b. The blue voxels in figure 9c are $IV$. The resulting array is the modified voxel array representing 'boundary' and 'inner' type voxel (figure 10), other voxels remain 'empty'. Results are shown in figures 12, 13 and 14.

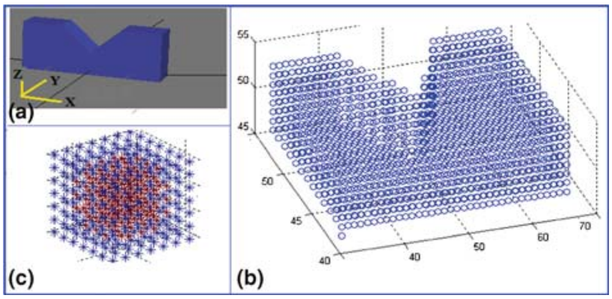


**Figure 12.** (**a**) STL coded object, (**b**) voxel conversion result, (**c**) cube object and neighbours formed around it.

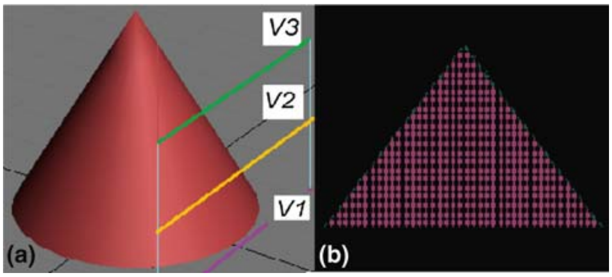


**Figure 13.** Voxelization results for a cone (**a**) shown as sections, (**b**) vertical section through apex.

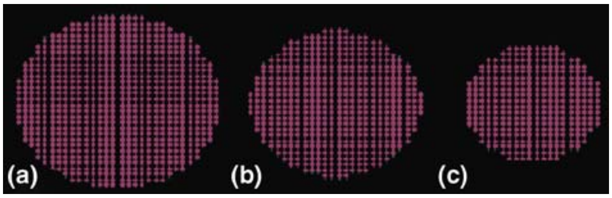


**Figure 14.** (**a**)–(**c**) show object voxels at various heights corresponding to $V1$–$V3$ starting from the base.

## 4.2 *A new algorithm for forming 'ranked neighbour'*

Let us imagine a small cube formed by $(2m + 1) \times (2m + 1) \times (2m + 1)$ voxels such that it has a centre voxel around which the axis is symmetric. This cube has a property that, when it is placed on the surface of an object, it tries to move towards the object centre till its central voxel coincides with a $BV$. In this state, all the voxels which are outside the object body but inside the cube can be classified as 'neighbours'. If all the $BV$s of a part are visited by the cube, then the entire object will have neighbour voxels which are of the order 1 (or rank 1). Subsequently, if the property of the cube is changed to react in a similar fashion, when it comes in contact with neighbour rank 1 voxel rather than surface voxel, then next layer (rank 2) can be formed. The process is detailed below to form an algorithm for voxelized 3D objects. Words 'rank' and 'order' have been used interchangeably.

Subspace of a cube around a single chosen object is formed from the 3D voxel array and referred to as the object voxel array $\mathrm{OVA}_{(x,y,z)}$ [33]. Its size is based on the extent of desired neighbourhood i.e., $q$ from the surface. All voxels belonging to other objects falling within this subspace are reset to 'empty' type.

The floating analyser zone (FAZ) is formed as 8 connected neighbouring voxel set around a centre voxel FZ($i$, $j$, $k$) (figure 15). It has $2m + 1$ layers of $(2m + 1) \times (2m + 1)$ voxel grids. This 3D zone is indexed by indices $i$, $j$, $k$ varying from $-m * i$ to $+m * i$, $-m * j$ to $+m * j$ and $-m * k$ to $+m * k$ along $x$, $y$ and $z$ directions, respectively. The size of FAZ is determined by integer $m$, around the centre voxel FZ($i = 0$, $j = 0, k = 0$). Object voxel array $\mathrm{OVA}_{(x,y,z)}$ is scanned. On finding a voxel with property as 'boundary', FZ is aligned with $\mathrm{FZ}_{(i=0,j=0,k=0)}$ at $\mathrm{OVA}_{(x,y,z)}$. A search within the local neighbourhood at $\mathrm{OVA}_{(x,y,z)}$ corresponding to FZ limits is carried out and for

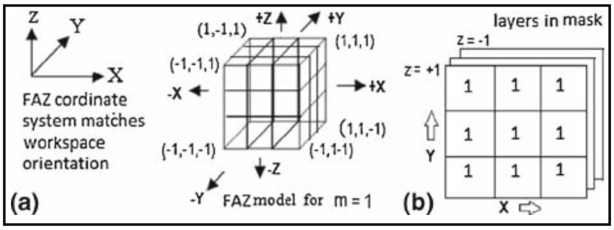


**Figure 15.** (**a**) Floating analyser zone and (**b**) mask values of SE.

a voxel with property 'empty' the type field is converted to 'boundary' and a parameter value '1' is assigned. For example, when FZ has $m = 1$ then all OVA$_{(x,y,z)}$ voxels with 'empty' property existing within the $(-1, -1, -1)$ to $(+1, +1, +1)$ limits of $i, j, k$ are changed (figure 16b). This process is continued over the entire $x, y, z$ index space of OVA$_{(x,y,z)}$. The resulting OVA with new 'neighbour' type in the type field and $n = 1$ in 'order' field is shown in figure 16a. Colour-coded 'inner' (blue) and 'boundary' (gray) voxels in one horizontal layer of the original OVA appears in figure 16b. After the above process, 'neighbour' order '1' (black) is also created (figure 16c). This completes one level of boundary grading.

For growing the graded neighbour zone, the previous process is repeated in phase 2 on processed OVA by searching for voxels in OVA-modified array data (figure 16a) that have 'neighbour' property and grade $n = 1$ and converting all 'empty' voxels within FZ to 'neighbour' grade 2. Repeating this modification on OVA for $n$ times creates voxelized object with $n$ graded neighbours surrounding it (figure 16c). All the objects in the workspace are treated similarly by forming their object voxel array OVAs and creating multigraded neighbour coding in them. The number of OVA thus created, will be equal to the number of objects in the workspace. These are mapped back into the 3D voxel array for the whole workspace.

## 5. Performance of the technique

Two aspects of the technique are tested for performance: (i) trueness of layer formation around the objects for holding viscosity association and (ii) viscosity seen by $\mathscr{R}$ and drag

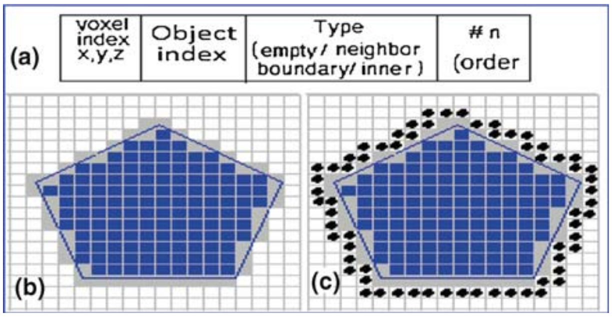


**Figure 16.** (**a**) Data record associated with a voxel, (**c**) neighbour processing results on object in (**b**).

generation for its movement in the neighbourhood. For easy appreciation of the result in printed form, they are presented in the form of planar slices from 3D voxelized spaces and neighbour of different order existing in voxelized space are shown in different colours.

## 5.1 *Fidelity of the layer-forming algorithm*

Voxelized layers formed by using FAZ of $5 \times 5 \times 5$, $7 \times 7 \times 7$ and $9 \times 9 \times 9$ (left to right) for circular (top) and generalized cylinders of complex shapes (figure 17).

Bigger FE is utilized in forming $\mu$ layers of higher thickness to cover a wide zone around the object with fewer layers. The outer layers show notable change in the shape of neighbourhood and their lack of trueness when compared to the original object (appearing as solid core at centre). This can be attributed to the FAZ shape that is only axis-symmetric on $X$, $Y$, $Z$ axes but not isosymmetric around $\mathrm{FZ}_{(0,0,0)}$.

5.1.1 *Improving layering algorithm.* In order to effectively shape FAZ in isosymmetric form, the FAZ is now modified to hold binary data at its cells (figure 15b). The value of $\mathrm{SE}_{(i,j,k)}$ may be 1or 0 for forming shaping element other than cube while maintaining high efficiency array-based ordered processing. A '0' inhibits processing at the location. The algorithm of the previous section is modified by permitting the boundary voxel conversion to 'neighbour' type and order '1' only, where the SE element value is '1'. For giving the SE a spherical shape, only those cells of SE that are within the radius of size $m$ from the SE centre $\mathrm{SE}_{(i,j,k=0,0,0)}$ are loaded with '1' value at the beginning of the process.

5.1.2 *Experimental results with spherical SE.* An FE having sphere shape was formed in $5 \times 5 \times 5$ cube by setting mask values as '1' in SE for voxels corresponding to the interior of the sphere of radius 2 voxels. Radius can be only integer values. Relative performance of the approach is shown on circular cylinder and complex cylinder for six layers (figure 18). In the first $\mu$ layer difference is not appreciable but as layering reaches

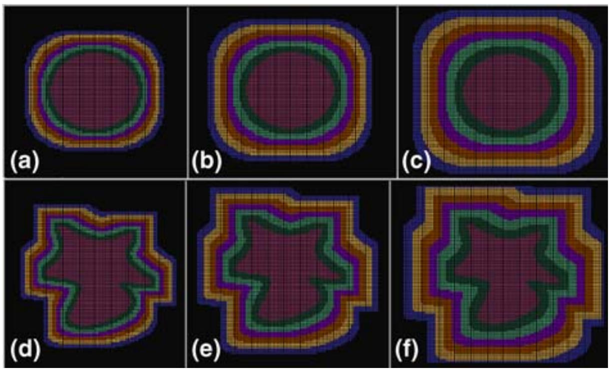


**Figure 17.** (**a**, **b**, **c**) (**d**, **e**, **f**) show the results of neighbour layer forming for circular and generalized complex section cylinders with surface concavity.

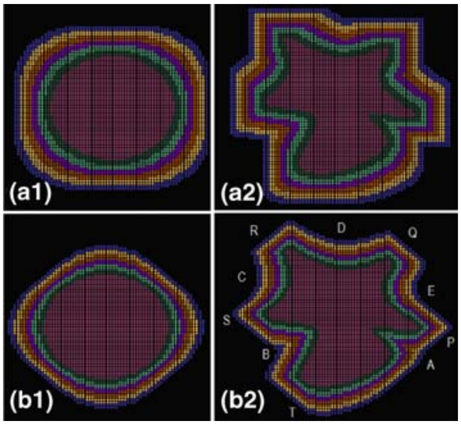


**Figure 18.** Relative performance of cube SE (**a1**), (**a2**) and sphere SE (**b1**), (**b2**) for $m = 2$ on circular and complex cylinder objects.

higher rank, improvement is amply clear. Narrow concavity at $A$ fills up thus preventing $\mathscr{R}$ from entering such areas while approaching a workspace part. Wider concavities like B, C, D, E shapes are maintained as these allow free space to remain intact for $\mathscr{R}$ to move freely in the workspace even with wide $\mu$ effected area around the object. Using $m = 3$ and 4, i.e., sphere contained by FAZ $7 \times 7 \times 7$ and $9 \times 9 \times 9$ improved layer formation up to six layers was achieved (figure 19). Refer table 1 for layering performance.

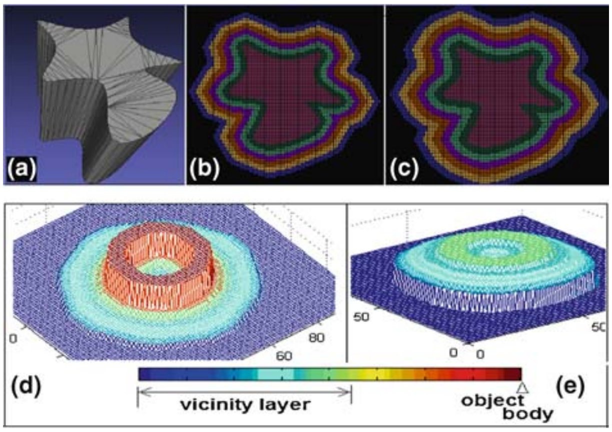


**Figure 19.** (**a**) Complex cylinder with concave surfaces, axis parallel to $z$, bottom planar surface at $Z = 20$ top surface at 60, (**b**) voxelization test results for spherical SE at the middle of the object height on complex section 3D cylinder, (**c**) SE formed in $m$=3 sized FAZ, (**d**) SE formed in $m$=4 sized FAZ. Voxelized ranked vicinity formed for hollow cylinder vicinity coding at the middle of the cylinder height. Red is the cylinder body, (**e**) result at $Z$ above top of cylinder. Voxels here are neighbour voxels from body below.

**Table 1.** Performance of layer-forming techniques.

| FAZ | SE | Size | Observations on layer fidelity |
|---|---|---|---|
| 3 | Cn3 | 27 | Layer size – thin, shape distortion in diagonal directions – high, redundant computation – low |
| 5 | Cn5 | 125 | Layer size – medium, distortion in diagonal directions – high, redundant computation – high |
| 7 | Cn7 | 343 | Layer size – wide, distortion in diagonal directions – high, redundant computation – higher |
| 3 | SR1 | <27 | Layer size – thin, shape distortion – moderate, % redundant computation – low. |
| 5 | SR2 | <125 | Layer size – medium, distortion – low, % redundant computation – lower |
| 7 | SR3 | <343 | Layer size – wide, shape fidelity and % redundant computation – lower than Case 5 |

Note: Shaping element $C_n$ – cube, $n$ is the size of the side in voxel, S is the approximated spherical shaping element, R is the radius of spherical shaping element.

## 5.2 *Performance of $\mu$ coded layers*

Experiments were carried on voxelized cone placed in workspace with vertex up. The spheres were formed in FAZ with $m = 2$ and 4 and six layers were formed (figure 20). Ranked neighbourhood was formed for the wedge (figure 12) by ranked layering around its voxel version (figure 12b) and the result is shown in figure 21.

The $\mathscr{R}$ moving in the vicinity of workspace part with ranked $\mu$ coded space develops a drag force as defined in (7). The ranked zones can have $\mu$ assigned to them as per a linear or nonlinear relation with the rank of the zone generally satisfying the concept

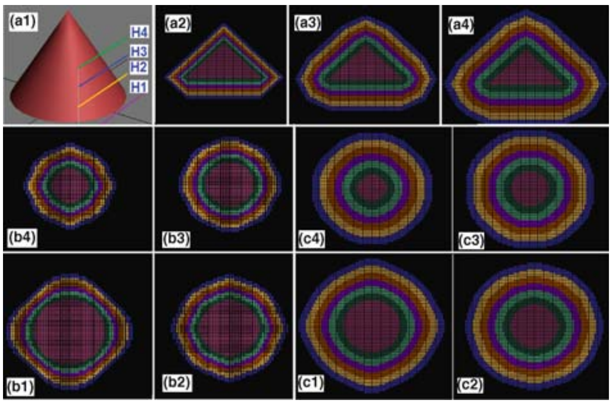


**Figure 20.** (**a1**) A cone object in workspace, (**a2–a4**) show vertical sections through voxelized layers formed by spherical SE in FAZ size $m = 1$, 3 and 4, (**b1–b4**): horizontal sections on H1($z = 20$) – H4($z = 40$) heights for spherical SE mask with $m = 2$, (**c1–c4**): horizontal sections on H1–H4 for spherical SE mask $m = 4$.

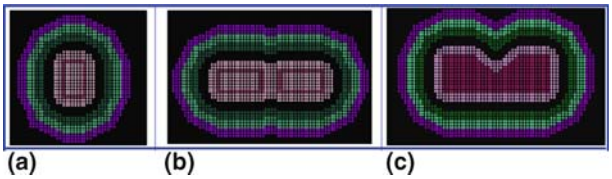


**Figure 21.** Results of seven-layer neighbour formed with sphere SE of radius 3 ($m = 3$) on wedge object of figures 12a, 12b and 12c as seen from $X$, $-Z$ and $Y$ directions, respectively. The view (**b**) shows the section at a height above the $V$ junction of the block .

developed earlier (figure 3). In an experiment, a cone (figure 20) was voxelized and its neighbourhood was formed with six ranked $\mu$ zones with $\mu_{(n)}$ as 32, 25, 20, 17, 15, 13 for $n = 1$–6, respectively.

The ranked $\mu$ zones have been formed by using the approximated sphere SE in FAZ with $m = 4$ as shown in figure 20 (c1, c2, c3, c4). It should be noted that the $\mu_{(n)}$ coding is a nonlinear function of the distance of the $\mu$ layer from the object surface. $\mathscr{R}$ with a unit surface area $S$ moves at a unit velocity on vectors like **Q**1, **Q**2, **Q**3 in the vertical plane which is located touching the cone at its base in a direction parallel to $y$-axis (figure 22). The results for point robot move are shown in figure 22 for six tracks parallel to $y$-axis at $Z = 1$ 0, 10, 20, 30, 40 and 45, respectively. Note that $\mathscr{R}$ moving on higher vector is farther from the cone surface and accordingly, there is reduction in drag force $\mathbf{F}_\mathrm{p}$. On track 6, which is at the level of the top node of the cone, the surface is away from the track of $\mathscr{R}$. No labelled voxel is encountered as the track passes outside the sixth $\mu$ layer. For complex shaped cylinder the drag $\mathbf{F}_\mathrm{p}$ is analysed by passing $\mathscr{R}$ through its neighbour with the unit velocity and assuming $\mu_{(n)}$ as 32, 25, 20, 17, 15, 13, respectively (figure 23).

The $\mathbf{F}_\mathrm{p}$ values shown by $\mathscr{R}$ are stepped in nature. The stepped effect is beneficial in some cases as the differential changes in force feedback is recognized efficiently by an operator. But the elimination of the effect is attempted if desired otherwise. The drag $\mathbf{F}_\mathrm{p}$ resulting from $\mu$ coding is velocity-dependant and is enhanced for faster moves (figure 24). The drag is accentuated more for faster moves in closer vicinity, a property that is highly desired.

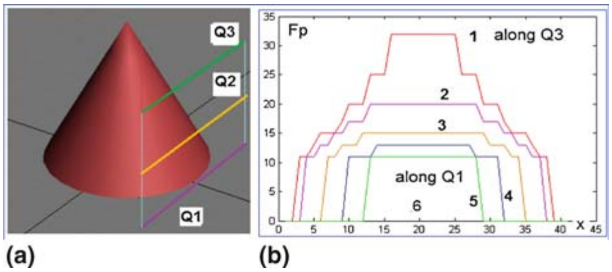


**Figure 22.** (**a**) A cone modelled in workspace and $\mathscr{R}$ moving in vertical plain of the vector **Q**s parallel to $x$-axis; 6 along **Q**1 and 1 along **Q**3; (**b**) $\mathbf{F}_\mathrm{p}$ faced by $\mathscr{R}$ along path 6 near the cone.

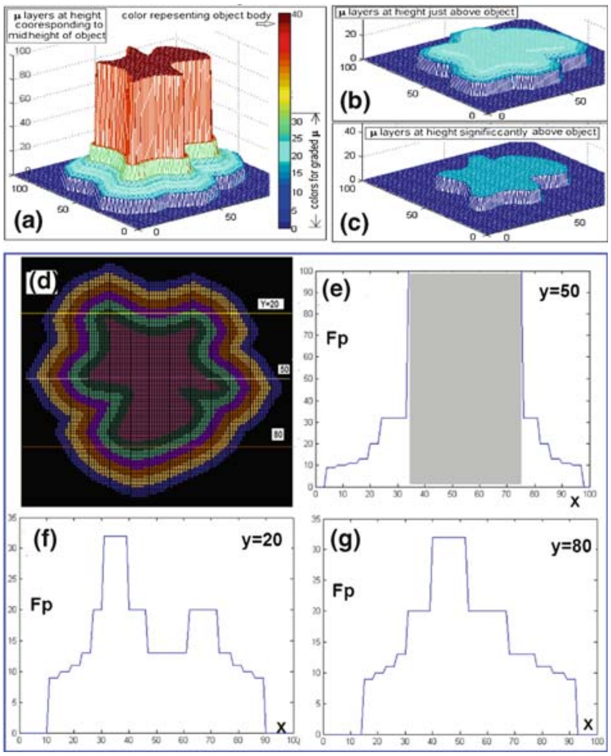


**Figure 23.** (**a**) $\mu$ assignment to vicinity at height matching the middle of the complex cylinder. Red mesh marking belongs to the object body, (**b**) result near the top of the object, (**c**) result of $\mu$ assignment at height higher than the top of the cylinder. Note: All voxels are of neighbour type only and the coded zone reduces at higher heights establishing full 3D behaviour. (**d**) Horizontal section ($z = 40$) of the complex profile cylinder with $\mu$ layers formed by sphere SE approximated in $9 \times 9 \times 9$ sized FAZ. The $\mu$ values in layers are 32, 20 13, 11, 10, 9, respectively in layers beginning near the surface which has a value 100, (**f**) $\mathbf{F}_\text{p}$ variation seen by $\mathscr{R}$ moving very close to the object but not touching it ($y = 20$) (mark the two peaks) (**g**) drag shown by $\mathscr{R}$ moving at $y = 80$, (**e**) drag on path at $y = 50$. Note: On contact with the body (shown as gray) it assumes high opposition owing to $\mu$ value corresponding to skin.

### 5.3 *Stepped $\mu$ change and its remedy*

The $F_\text{p}$ values shown by $\mathscr{R}$ are stepped in nature owing to the layers having multiple voxel width. Single voxel wide $\mu$ layers can give smooth $\mathbf{F}_\text{p}$ but is not permissible due to the resulting layer shape fidelity loss. The obvious option of reducing voxel size leads to data explosion conditions. A technique has been developed to produce relatively smoother $\mathbf{F}_\text{p}$ values without increasing data size. A circular memory buffer is formed between high limit (HL) and low limit (LL) (figure 25). It holds latest $k$ encountered $\mu$ values. During a move, whenever $\mathscr{R}$ encounters a voxel with $x$, $y$, $z$, indices different from the previous voxel (i.e., a new voxel) its $\mu$ is recorded using the buffer write pointer (BWP) and BWP

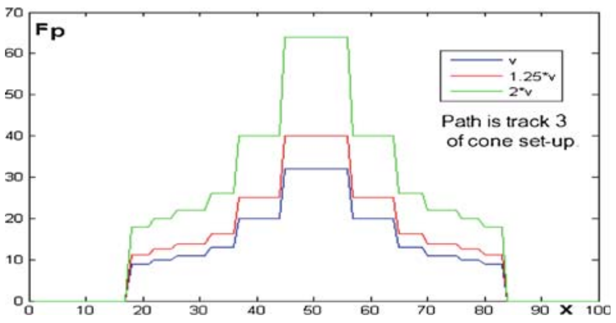


**Figure 24.** $\mathbf{F}_\mathrm{p}$ shown by $\mathscr{R}$ along a straight path near the cone at different speeds.

is advanced. Only one buffer is used for each $\mathscr{R}$. The maximum buffer depth $u \geq (2 * m)$. Here $m$ corresponds to the biggest FAZ used for any object in the Cartesian workspace. This ensures that in an implementation with the varied $m$ of FAZ for different objects too, the buffer is sufficient for $\mu_\varepsilon$ computation using

$$\mu_\varepsilon = \frac{1}{k} * \sum_{i=0}^{i=k} \mu_{(\mathrm{BWP}-i)} \tag{6}$$

Note that only $k$ data are used from the current value backwards and higher depth of buffer does not affect the method. The resultant 'drag' profiles for the modified algorithm's result are shown in figures 26 and 27.

Shaping of the filter can be carried out by assigning weights to the previous $\mu$ values in the following manner:

$$\mu_\varepsilon = \sum_{i=0}^{i=k} \alpha_k * \mu_{(\mathrm{BWP}-i)} / \sum_{i=0}^{i=k} \alpha_k. \tag{7}$$

Experiments have been carried with $k = 1$, 3 and 10, for constant $\alpha_\iota$ as well as by varying $\alpha_\iota$ for different $i$. Two sets of experiments for linear as well as nonlinear $\mu$ coding over the ranked neighbour layers are analysed. The test cases are given as follows (for results, refer figures 26–30):

Case-Ia – no memory ($k = 1, a = 1$), linear $\mu$ coding.
Case-IIa – long sustained memory ($k = 10, \alpha_1$–$\alpha_{10} = 1$), linear $\mu$ coding.
Case-IIIa – fading memory ($k = 10, \alpha_1$–$\alpha_{10} = 1$, 0.9, 0.7, 0.5, 0.55, 0.30, 0.2, 0.1, 0.05, 0.02), linear $\mu$ coding.

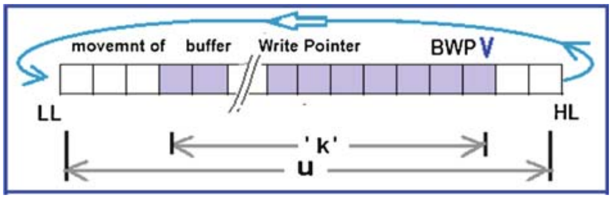


**Figure 25.** Circular buffer for recording latest $\mu$ values.

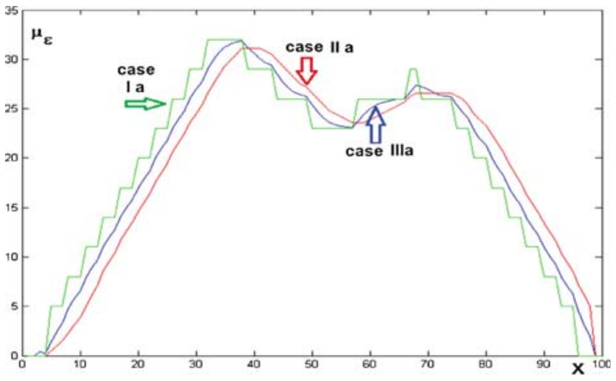


**Figure 26.** Results of traverse of $\mathscr{R}$ in linear $\mu$ coded ranked neighbour (for case in figure 23, $y = 20$) and smoothing by filter definition Ia, IIa, IIIa. (Refer table 2, the first 3 rows.)

Case-IVa – short sustained memory ($k = 3, \alpha_\iota = 1, 1, 1$), linear $\mu$ coding.
Case-Ib – no memory ($k = 1, \alpha = 1$), nonlinear $\mu$ coding.
Case-IIb – long sustained memory ($k = 10, \alpha_1$–$\alpha_{10} = 1$), nonlinear $\mu$ coding.
Case-IIIb – fading memory ($k = 10, \alpha_\iota$–$\alpha_{10} = 1, 0.9, 0.7, 0.5, 0.55, 0.30, 0.2, 0.1, 0.05, 0.02$), nonlinear $\mu$ coding.
Case-IVb – short sustained memory ($k = 3, \alpha_\iota = 1, 1, 1$), nonlinear $\mu$ coding.

## 5.4 *Observations on linear and nonlinear $\mu$ codings*

The drag performance of the viscosity emulation by linear and nonlinear codings vary and depend on memory (table 2). The distance to the object is different for the two locations on $y = 20$ in figure 23 and form peaks of drag but the drag is enhanced by nonlinear coding of the neighbour viscosity (figure 30). Note the distinct rapid rise in the case of

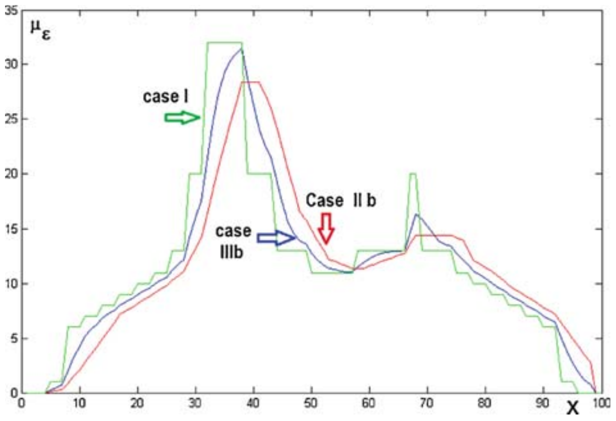


**Figure 27.** Results of traverse of $\mathscr{R}$ in linear $\mu$ coded ranked neighbour (case in figure 23, $y = 20$) and smoothing by Ib, IIb, IIIb. (Refer table 2, fifth, sixth and seventh rows.)

**Table 2.** $\mu_e$ performance with different memory.

| $\mu$ | Case | Effective $\mu$ behaviour |
|---|---|---|
| I | Ia | Large steps of $\mu_e$, gradual increase closer to the object. Operator feel jerky. |
| I | IIa | Smooth $\mu_e$ gradual increase closer to the object, smooth drag generation, delay large, may miss narrow features |
| I | IIIa | Smooth $\mu_e$, gradual increase closer to object, feel to operator smooth. Response delay moderate |
| I | IVa | Fine-stepped $\mu_e$, gradual increase closer to object, feel to the operator fine -like notch move, easy to feel, no delay in response (figure 28) |
| II | Ib | Large steps of $\mu_e$, steep increase closer to the object, vicinity change too jerky |
| II | IIb | Smooth $\mu_e$, gradual increase closer to object, smooth drag generation, delay large, may miss narrow features |
| II | IIIb | Smooth $\mu_e$, progressive rate increase closer to the object, force feel to operator smooth and closeness dependent rate of change, response delay moderate |
| II | IVb | Fine-stepped $\mu_e$, develops comb rub feel, progressive increase of rise rate closer to object, operator feels fine - taps feel like rubbing on comb-teeth, easy to feel degree of closeness owing to increasing magnitude difference at successive voxels, no delay (figure 29) |

Note: $\mu$ coding type I linear (32, 29, 26, 23, 20, 17, 14, 11, 8, 5). Type II nonlinear (32,20,13,11,10,9,8,0,0,0,0). Cases are as defined based on memory property.

nonlinear coding. It also results in energy saving during the traverse of $\mathscr{R}$ on the path. The energy saving is proportional to the area between the respective drag plots of the memory-based types in two sets of curves for the linear and nonlinear $\mu$ codings.

## 6. Perception enhancement by viscosity emulation

The previous sections establish the effectiveness of the voxelized and $\mu$-activated workspace model for emulating graded (static) viscous layers and their drag effects on $\mathscr{R}$ for a known velocity. Extending the model by placing a kinematics equivalent slave arm in the same environment (figure 31) and matching its instantaneous spatial state to

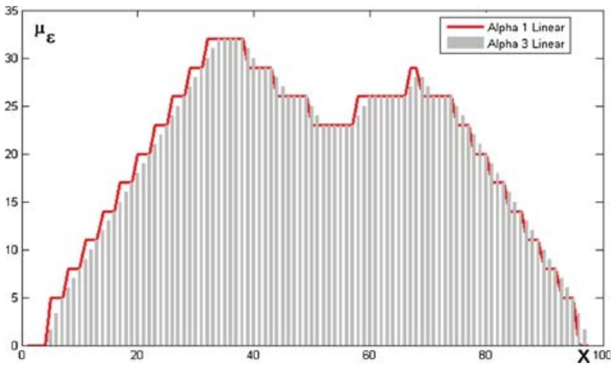


**Figure 28.** Relative performance for stepped and fine-stepped $\mu_\varepsilon$ for Cases Ia and IVa for linear $\mu$ coding (32, 29, 26, 23, 20, 17, 14, 11, 8, 5). The fine-stepped responses (gray bars) are very close to the original values and produce comb rub feel. However, the drag rise rate is constant.

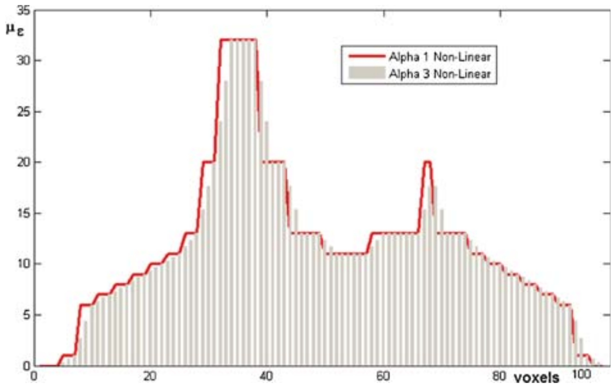


**Figure 29.** Relative performance in stepped and fine-stepped $\mu_\varepsilon$ for Cases Ib (red) and IVb (gray bars) with nonlinear $\mu$ coding ($\mu_1$–$\mu_6$ are –32, 20, 13, 11, 10, 9). The fine-stepped responses (gray bars) are very close to the original values and produce comb rub feel. Drag build-up near the object is steep.

that of slave, subjects an attached point on the robot model to viscous drag environment in equivalent form. This point represents $\mathscr{R}$. The emulated viscous drag effect for any point on the robot body and its reflection on the master can be computed by the method outlined in figure 32. Torque transducers on robot joints can realize it in physical form.

Consider a single DOF arrangement as depicted in figure 33. The drag is introduced in force balance arrangement in master as

$$\mathbf{M}x'' + \mathbf{B}x' + \mathbf{D}x' + \left(\frac{1}{\mathbf{A}}\right)\mathbf{F}_e = \mathbf{F}_h. \tag{8}$$

In figure 33, $\tau_1$ represents the synthesized drag term which is denoted as

$$\tau_1 = \mu_\varepsilon * \mathbf{V} \tag{9}$$

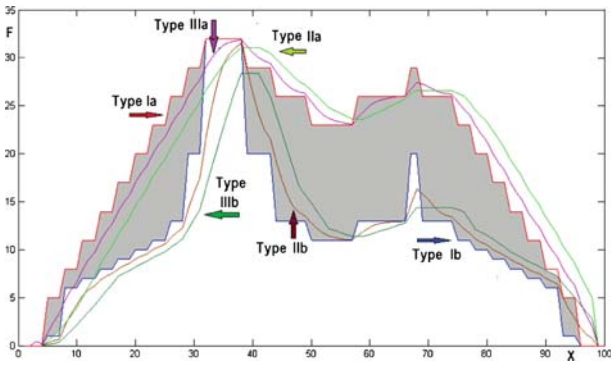


**Figure 30.** Comparative performance of linear (upper cluster) and nonlinear viscosity coding (lower cluster) of the ranked neighbour layers. Note the distinct rapid rise in the case of nonlinear coding and energy saving (shaded area).

and $\tau_2$ represents the $\mathbf{F}_e$ effects from the slave to the master as feedback and accounts for the inertia and friction terms of the slave arm. $V$ is computed by the real-time processing block (figure 32). $G$ and $A$ are gain and attenuation for controlling their effects. $L$ is the distance of force application point from the joint axis. Using λ for torque constant of the actuator in figure 33 gives

$$\mathbf{M}x'' + \mathbf{B}x' + ((\tau_1 * G + (1/A)\,\tau_2)\,/L) * \lambda = \mathbf{F}_h. \qquad (10)$$

The viscous drag-based vicinity effect emulation has two outcomes: (1) it slows down the master arm and thus, slows down the slave arm in the vicinity of the object body and (2) the opposition to operator arm action forms a tactile sense in the hands of the operator.

When the slave is not in contact with any object, $\tau_2$ is negligible. The effort-absorption by master arm's physical property $M_t$ and $B_t$ are very low for light and balanced master arm operating at moderate speed. Hence, $\tau_1$ is the dominant signal. Consequently, the

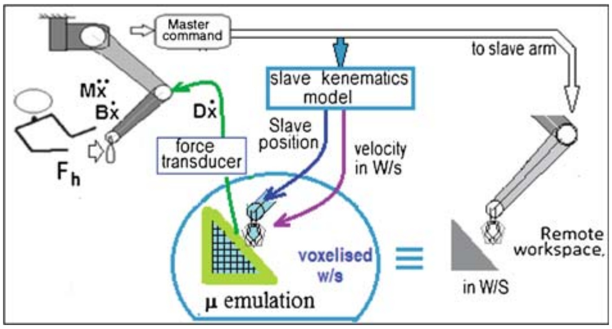


**Figure 31.** Modality of operator aid using the physics-based model in overall tele-operation. The viscosity emulation effect is perceived by the operator as an opposing drag.

viscosity-based drag is a dominant effect and is easily perceived by the operator. Figure 34 shows the force absorbed by the viscosity emulation at different speeds of moves along $x$-axis, near the cone object (figure 22) when the ranked viscosity is nonlinear in nature. For $X$ corresponding to the dotted line pair, the $\mathscr{R}$ is nearest to the object and absorption in viscous layer also, is the highest. The lower dark shade shows part of the operator's effort available for master move.

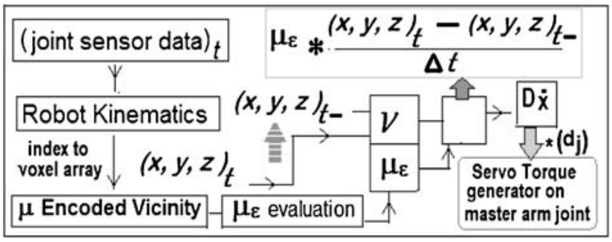


**Figure 32.** On-line computation scheme for determining $\mu_\varepsilon$ and $\mathscr{R}$'s velocity at time $t$ to form $D$. The method is applicable for any point on the robot body by implementing the kinematics model till the point along the robot arm.

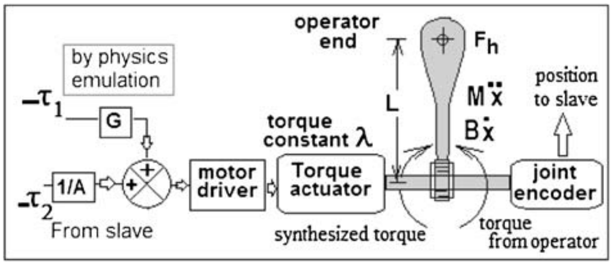


**Figure 33.** Viscosity-emulated opposing drag integration for a single DOF with force feedback from the slave. Note: the two rotating arrows are in opposite directions for feedback and operator input actions.

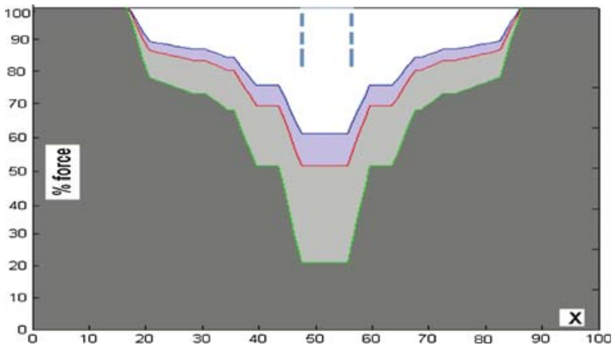


**Figure 34.** Force from human operator reaching master side arm. Dark shades show the forces absorbed by the viscosity emulation at different speed of approach.

Higher the approach velocity, higher will be the absorption in $\mu$ layers. Also the absorption increases near the surface of the object. Voxel-to-space map is fixed for the entire cubical workspace as per the desired scale factor (i.e., grid spacing). The voxelized workspace (figure 31) is formed by a 3D data array of equal voxel size as shown in figure 10. The joint value and kinematics model give the locations $X$, $Y$, $Z$ of $\mathscr{R}$ [34]. Voxel-to-space map gives voxel 3D index into the data record. The applicable $\mu$ is accessed from the record directly at real-time speed and $\mu_\varepsilon$ is computed without any constraint. Time-stamped position record or the joint velocity sensors provide velocity as real-time acquired data (figure 32). The figure shows computation steps for one point attached to slave robot model which is repeated for all the points (they encounter different $\mu_\varepsilon$ and move at different speed $V$) attached to the slave robot model. The conversion to real physical force is carried out by using servo motors in the master arm [35] as indicated earlier (figure 33). The engineering implementation is discussed in another work.

## 7. Conclusion

The viscous drag emulation process in object vicinity, needs precise spatial modelling of the vicinity of complex surfaces of real objects. Real-time sensing of the local viscosity property assigned to a location, where point of consideration on robot body passes, must be accurate and fast. Modelling of precise shape of the real workspace object is of utmost importance and necessitates the use of accurate CAD models [36] for assigning physical property. Unlike the potential maps for mobile robots that are approximate the viscosity assignment is precise.

The voxelized viscosity layer forming technique developed in this study and the varying viscocity emulation are the key to the success of developing a vicinity perception in real-time. This method is efficient and amenable to automated viscous layer model formation from CAD surface models without any shape constraints. The fidelity of ranked layers is good when spherical-forming-elements of moderate and large sizes are used. The consequent accentuation of stepped drag effect is also solved by using filters. The ranked viscosity model shifts most of the computations to offline preprocessing phase and thus, the real-time computation burden is minimized. The dependence of the drag phenomena on robot velocity and nearness to the object surface make the technique a strong candidate for force absorption-based active interface in ‘man-in-loop’ telecontrol of remote manipulators. The perception developed by physics emulation is a novel technique and functions successfully in precontact phase for forward position force feedback modality of teleoperation.

## Acknowledgements

The author thanks Dr L M Gantayet and Dr B K Datta for useful discussions.

## References

[1] H Kazerooni, T-I Tsay and H Karin, *IEEE Trans. Control Syst. Technol.* **1(1)**, 50 (1993)

[2] D A Lawrence, *IEEE Trans. Robot. Autom.* **9(5)**, 624 (1993)

[3] J E Speich, K Fite and M Goldfarb, *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 2671–2676 (2000)

[4] M C Cavusoglu, A Sherman and F Tendick, *IEEE Trans. Robot. Autom.* **18**(**4**), 641 (2002)

[5] Z Zhang, D Zhao and T Chen, *IEEE International Conference on Robotics and Biomimetics (ROBIO 2007)*, pp. 307–311 (2007)

[6] A M Abeykoon and K Ohnishi, *IEEE International Conference on Industrial Informatics*, pp. 833–838 (2006)

[7] J K Mukherjee, U Mahapatra, S Bhattacharya and G P Srivastava, *Proceedings of the International Conference on Peaceful Uses of Atomic Energy-2009*, *V. 2* 2009

[8] J K Mukherjee and K Y V Krishna, Machine Intelligence Tools For Supporting Metrics on Active Structural Elements and On-line Inspection in Fuel Cycle Facilities, *International Conference SMiRT21* (New Delhi, India, 2011)

[9] G R Hopkinson, S Alexander, V Cyril, Z Igor and D H Andrew, *IEEE Trans. Nucl. Sci.* **52**(**6**), 2664 (2005)

[10] S Payandeh and Z Stanisic, *Proceedings of the 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, *HAPTICS 2002*, IEEE, 2002, pp. 18–23

[11] S Park, R D Howe and D F Torchiana, Virtual fixtures for robotic cardiac surgery, *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2001* (Springer, Berlin, Heidelberg, 2001) pp. 1419–1420

[12] J K Mukherjee, An observer based safety implementation in tele-robotics, *in: Proceedings of the International Conference TIMA-2011*, 2011

[13] C W Warren, *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 316–321 (1989)

[14] Y Koren and J Borenstein, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1398–1404 (1991)

[15] E Sanchez, A Rubio and A Avello, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, pp. 1302–1307 (2002)

[16] B Hannaford and J H Ryu, *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1863–1869 (2001)

[17] C O Bennett and J E Myres, *Momentum, heat and mass transfer* (McGraw Hill, New York, 1962)

[18] V Chandru, S Manohar and C E Prakash, *IEEE Comput. Graph. Appl.* **15**(**6**), 42 (1995)

[19] E G Gilbert, D W Johnson and S S Keerthi, *IEEE J. Robot. Autom.* **4**(**2**), 193 (1988)

[20] J O Rourke and N Badler, *IEEE Trans. Pattern. Anal. Mach. Intell.* **3**, 295 (1979)

[21] J Tomero, J Hamlin and R Kelley, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1602–1608 (1991)

[22] K K Gupta, *IEEE Trans. Robot. Autom.* **6**(**5**), 522 (1990)

[23] C K Chua, K F Leong and C-S Lim, *Rapid prototyping: Principles and applications* (World Scientific Publishing Co., Inc., 2003)

[24] A Kaufman, D Cohen and R Yagel, *IEEE Computer* **26**(**7**), 51 (1993)

[25] A Kaufman and E Shimony, *Proceedings of the 1986 Workshop on Interactive 3D Graphics*, pp. 45–75

[26] O Cohen and A Kaufman, *Graphical models and Image Processing* **57**(**6**), 453 (1995)

[27] J K Mukherjee, *Conference on Convergent Technologies for the Asia-Pacific Region, TENCON 2003*, Vol. 1, pp. 305–309

[28] O Cohen and A Kaufman, *IEEE Comput. Graph. Appl.* **17**(**6**), 80 (1997)

[29] J Huang, R Yagel, V Filippov and Y Kurzion, *IEEE Symposium on Volume Visualization*, pp. 119–126 (1988)

[30] T Moller *J. Graphics Tools* **2**(**2**), 25 (1997)

[31] O Devillers and P Guigue, Faster triangle-triangle intersection tests, Technical Report 4488, INRIA, 2002
[32] H M Erit, *J. Graphics Tools* **2**(**2**), 25 (1997)
[33] H Samel, *Design and analysis of spatial data structures in computer graphics, image processing and GIS* (Addison Wesley, Reading, MA, 1990) Vol. 85
[34] M W Spong, S Huthinson and M Vidysagar, *Robot modeling and control* (Wiley, New York, 2006) Vol. 3
[35] Y Huang, G Zhao, G Bao and Z Wang, *IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, Vol. 1, pp. 682–686 (2012)
[36] A van Dam, S K Feiner, J F Hughes and R L Phillips, *Introduction to computer graphics* (Addison Wesley, Reading, 1994) Vol. 55