Modeling of geometric correspondences and transformations via geometry constructions language

A A Boykov
MIREA – Russian Technological University, Moscow, Russia

E-mail: albophx@mail.ru

Abstract. Geometric correspondences and transformations can be applied in solutions to numerous problems of geometric modeling, mainly in order to create models of curves and surfaces. Among the tools of CAD systems, the geometric transformations are presented in the form of a limited set of editing tools, array tools, tools providing formation of 3D solids and surfaces through motion. In practice, this set of tools restricts application of the wide capabilities of geometric transformations and correspondences. In this regard, it is proposed to model the geometric transformations and correspondences via the geometry construction language. The elements of the geometric transformation theory are analyzed. The geometric transformations are proposed to be created on par with other objects of the geometric model. The commands of the geometry constructions language enabling construction of the transformation objects are considered, their application is discussed. Solutions to practical problems of surface modeling via the geometry constructions language with the use of geometric transformation are demonstrated on examples. The proposed approach allows us to facilitate modeling of curves and surfaces in CAD systems, if said curves and surfaces can be acquired through geometric transformation or correspondence that can be defined via a certain constructive algorithm.

1. Introduction
Geometric correspondences and geometric transformations in particular find wide application in modeling of curves and surfaces. For example, geometric correspondences can be applied in geometric modeling of moving parts of machinery, physical phenomena (e.g. reflection), cutting tool surfaces, architectural shell surfaces etc. In CAD systems the geometric transformations are realized as a part of the basic editing tools (copy, rotate, scale, stretch), array tools, formations tools (extrude, revolve, sweep). A closer look reveals that only a handful of the possible geometric transformations have been realized in CAD systems (displacement, similarity, affinity). Application of other types of geometric transformations or correspondences is virtually impossible in CAD systems since geometric transformations and correspondences are not considered independent geometric objects even though they constitute independent geometric objects from the point of view of geometric modeling in general.

In the present paper the approach to geometric modeling of correspondences and transformations through the previously considered Geometry Constructions (GC) language is proposed. The application of the geometric models of correspondences and transformations in solutions to certain tasks of computer geometric modeling is considered.
2. Problem definition

The term “geometric transformation” is fundamental in modern geometry, since various divisions of geometry correspond to certain classes of geometric transformations [1, 2]. Let us list some of them:

- displacements;
- similarities;
- affine transformations;
- projective transformations;
- circular transformations, etc.

The above transformation classes are listed in the order of increasing generality – thus the similarities contain the displacements, the affine transformations contain similarities; the classes of projective transformations and circular transformations are independent and belong to a more general class of birational transformations, and so on.

The CAD systems realize the transformations of offset (translation), turn (rotation) and flip (axial symmetry) of the displacements class, scaling and shear of the affine transformations class. Through this limited set of transformations the basic formation operations – extrusion (translation), revolution (turn), motion following a path (a combination of translations and turns) are realized. Obviously, the vast majority of geometric transformations are not realized through the tools of CAD systems.

Their realization, as it was already mentioned, can be performed manually through numerous typical commands or programmed. In paper [3] a specific programming language GC designed for automation of geometric construction is proposed.

In order to realize the geometric transformations and correspondences via the proposed programming language for geometric construction, it is required:

- to consider the nature of geometric correspondences and transformations in application to geometric modeling in CAD systems;
- to define the corresponding elements, extend the syntax of the GC language and to provide guidelines to its usage;
- to consider the application of the GC language to modeling of geometric correspondences and transformations and their practical application.

Let's discuss the ways to meet these objectives.

3. Theory

The scope of the present paper is restricted to two-dimensional geometric correspondences and transformations; however, the proposed approach can be applied to geometric constructions in space as well.

Our main concern is the functional and bijective correspondences, as well as the particular case of geometric bijections called transformations. Geometric correspondences can be classified into point correspondences, non-point correspondences, and correlations that bind figures of different types.

A correspondence can be defined analytically (through a system of equations) or constructively. In the latter case, the objects of the initial multitude (the initial figures) and the objects of the target multitude (the corresponding or resulting figures) are connected by a certain geometric construction. Any geometric correspondence can be defined as a set of arguments – numeric or geometric (figures). The constructive approach has the advantage of not requiring the user to input the analytic dependencies and calculate the transformation parameters based on the parameters of the initial shapes.

It follows from the above that, in essence, the geometric correspondence or transformation constitutes an object quite similar to common shapes – for example, just as a straight line is defined by two points, the translation transformation is fully defined the initial and the final point. Therefore, the transformations in the GC language are created the same way as common figures:

\[ \text{identifier}_t = \text{command}_t (\text{arguments}), \]

where \( \text{identifier}_t \) is the internal name for the transformation object, and \( \text{command}_t \) is the name of the geometric command creating the object.

In constructive approach, it is more essential to pass figures as arguments. For example, a turn within a plane can be defined with the center (a point parameter) and the value of angle (a numeric parameter), etc.
however in practice it is more natural to specify the center of the turn and a pair of points defining the angle, in which case one does not have to measure angle, express it in numeric form, round it, etc. In this regard, we are going to apply the following approach to specifying the arguments:
– for standard correspondences and transformations we are going to specify a group of stationary (fixed) and two groups of corresponding figures, if there are no stationary figures – only two groups of corresponding figures;
– for involutions we are going to specify a stationary figure or a group of stationary figures.
In order to group the arguments, let us use the character “:”:
transformation1 = command1 (stationary figures : initial figures : correspondent figures)
Table 1 lists the commands of the GC language used to create standard transformation objects.
The application of the resulting transformation object does not depend on the standard transformation used to create it.
The direct transformation is performed with the command trobj:
[t1, t2, ...] = trobj (transformationObject, o1, o2, ...).
The inverse transformation is performed with the command trobi:
{o1, o2, ...} = trobi (transformationObject, t1, t2, ...)
Along with object transformations, the programs may include rapid inline transformations leading to a more compact code. The inline transformations are translate, rotate and reflect:
[t1, t2, ...] = translate (dx, dy: o1, o2, ...)
[t1, t2, ...] = rotate (P, angle: o1, o2, ...)
[t1, t2, ...] = reflect (Center-or-axis: o1, o2, ...)
Let us consider in detail the particularities of practical application of standard transformations.
Linear transformations, from displacement to affinity, work within a limited area of the modeled space close to the origin of coordinates. Let us call this area “the area of finite geometry”. When a certain figure is transformed, a new figure within the same area is acquired; we do not have to deal with the elements falling into the infinitely distant areas. In addition, these transformations rather carefully preserve the shape of the original figures – points, lines, and even Bezier splines remain themselves. A circle might be transformed into an ellipse and vice versa, but both figures, as a rule, remain standard. For this reason, these classes of transformations are widely applied in CAD systems.
The situation with other classes of transformation – projectivity and inversion – is on the contrary. Projective transformation deals with projective plane that includes infinitely distant elements that can take finite values upon transformation and vice versa. A circle can be split and become a parabola or a hyperbola, the same as any figure (e.g. a spline segment) can be stretched into infinity. Closed curves passing through the center of inversion are split, while infinite lines are folded into closed ones. Complications arise with the figures as well: while straight lines remain straight, second-degree curves can transform into each other. This requires the CAD system to support parabolas and hyperbolas, the same is true for other curves that can be a result of transformation of spline segments. The inversion is a quadratic transformation that in general doubles the order of the algebraic curves. A straight line becomes a circle, an ellipse becomes a fourth degree curve.
It is therefore vital to answer the following essential questions in order to create a geometric system realizing other transformations as well as the affine transformations:
– How does one treat infinitely distant points including ones appearing within a figure, and infinite objects in general?
– How does one represent figures not included into the standard objects of a common CAD system?
Restriction 1. Elements of finite geometry (elements that are involved directly in formation) cannot contain infinitely distant points. These are the figures returned by the geometric program. Otherwise the execution is halted and an exception message is displayed. This restriction returns us into the area of finite geometry where affine transformations can function.
Restriction 2. Elements of supplemental geometry can contain infinitely distant points that are represented, as it is common, in the form of homogeneous coordinates, however, such elements cannot take part in formation process and are used exclusively in intermediate construction (concealed in visualization).
Restriction 3. Any non-standard figure (standard figures are points, straight lines and line segments, circles and arcs, ellipses) have to be converted into a general figure (replaced with a discrete set of points – common and infinitely distant). In visualization and formation the general figures are replaced with splines.

Table 1. GC language commands for standard transformations.

| Command | Description | Usage |
|---------|-------------|-------|
| trofs   | Offset (Figure 1(a)) | translationTransformation = trofs (P_0 : Q_0) |
| trflip  | Central and axial symmetry (Figure 1(b)) | flipTransformation = trflip (pointOrStraightLine) |
| trmv    | Displacement (Figure 1(c)) | displacementTransformation = trmv (P_1, P_2 : Q_1, Q_2) |
| trsim   | Displacement with stretch (similarity without symmetry) (Figure 1(d)) | similarityTransformation = trsim (P_1, P_2 : Q_1, Q_2) |
| trafl   | Affine transformation of general form (Figure 1(e)) | affineTransformation = trafl (P_1, P_2, P_3 : Q_1, Q_2, Q_3) |
| trafx   | Axial affine transformation (Figure 1(f)) and homothety (Figure 1(g)) | axialAffineTransformation = trafx (axis : P_0 : Q_0) |
| trpr    | Projective transformation of general form (Figure 1(h)) | projectiveTransformation = trpr (P_1, P_2, P_3, P_4 : Q_1, Q_2, Q_3, Q_4) |
| trhom   | Homology (Figure 1(i)) | homologyTransformation = trhom (center, axis : P_1 : Q_1) |
| trc2p   | Carthesian coordinate system to polar coordinate system transformation (Figure 1(j)) | carthesian2polarTransformation = trc2p (O_d, W, H : O_p, R) |
| trinv   | Inversion at a sphere cF (Figure 1(k)) | inversionTransformation = trinv (circle) |
| trcomp  | Composition of transformations | transfComposition = trcomp (T_1, T_2, …) |

If construction requires an infinite figure (e.g. a straight line) as a whole to be replaced with a discrete set of points, the following method can be applied (Figure 2(a)). One can pick a fitting bijective correspondence with a closed curve (e.g. a circle), replace the closed curve with a discrete set of points and transform it into the desired set of points of the infinite figure.

It should be noted that calculation accuracy for the points outside the finite geometry area can be relatively low, therefore for each of such cases additional tweaking of the number of discrete points, the order of execution, etc. might be required.

With regard to the introduced restrictions, let us consider the modeling of custom transformations. Any transformation can be replaced with a pair of constructions realizing direct and inverse transformation of figures. Such transformation object can be created the following way:

```
customTransformation = trcreate (classes, subtrobj, subtrobi)
customInvolution = trcreate (classes, subtrobj)
```

where classes are the types of standard objects not required to be converted into general figures for the transformation; subtrobj and subtrobi are previously defined constructions realizing direct and inverse transformation. The transformation procedures have to realize at the very least transformation of a point, in that case every other standard object will be converted into a general figure.
Figure 1. Standard geometric transformations.

Mapping, unlike transformation, is irreversible in most cases, and does not require the inverse procedure. According to the GC language specifics [3], it is not required to create a transformation object, since this function is performed by the transformation function. Regardless, in order to provide flexible control over discretization of the visualized curves, it is possible to use the command for discrete set formation:

$$\text{mappedObject} = \text{prow} (\text{mappingSub}, \text{sourceObject} : \text{parameters})$$

– where \text{mappingSub} is the constructing procedure of the mapped point, \text{sourceObject} is the initial line that is subject to mapping, \text{parameters} is a set of arbitrary parameters controlling discretization of the initial curve (number of points, the initial point, direction of execution).

Finally, let us briefly consider the correspondences of general form – the multivalued correspondences. Multivalued correspondence modeling is realized through a construction procedure returning a number of mapped objects. For example, a model of a sphere on a drawing establishes a 1-2-valued correspondence between its projections (Figure 2(b)). The main complexity here is to regulate the order of the constructed points that conserves continuity in the multitude of solutions for various initial values of the parameters. In general, errors may occur, for example, Figure 2 (c) depicts incorrect point order, while Figure 2(d) illustrates the correct point order. The main method allowing to achieve continuity in cyclic construction is to sort the solutions by proximity to the solutions acquired at the previous step. This is the function of the utility character “#”:

$$[Q_1, Q_2, ...] = \text{customCorrespondence} (P)$$  
// unsorted objects
Directive «#» instructs the interpreter to cache the result of execution of this instance of construction procedure and to apply it further in order to sort the subsequent results. This method does not eliminate erroneous sorting entirely, especially when the trajectories of the solutions intersect, but it proved sufficient in many cases. A more in-depth study of connection between the constructive solutions to geometric tasks on construction and their respective analytic counterparts allowing for precise classification of solution variants is a separate urgent problem beyond the scope of the present study.

Figure 2. Particularities of geometric transformations and correspondences modeling.

Summing up, the GC language allows one to create models of geometric correspondences and transformations on par with other geometric objects, such as points and lines. This allows one to operate on sets of correspondences and transformations as on sets of figures – bundles, arrays, etc. This, in turn, offers wide opportunities in modeling of so-called stratified transformations and correspondences, that constitute correspondences of higher order that can be reduced to more simple, in particular, two-dimensional, correspondences. The basic means of the GC language allowing for automated creation and application of models of geometric transformations and correspondences are documented above.

4. Results of experiments

Let us consider modeling of transformations and correspondences via the GC language on practice. As in [3], let us only perform two-dimensional constructions.

Example 1. Paper [3] considered construction of an element of an onion-shaped dome consisting of two segments of Dupin cyclides. It is known that many Dupin cyclides can be acquired through inversion of second-degree surfaces [4]. Let us consider construction of a model of a dome element through the inversion transformation. It is worth noting that spatial inversion at a sphere within a bundle of planes passing through the center of the sphere is reduced to planar inversions, therefore this transformation can be readily realized using only the methods of descriptive geometry.

Let us assume outlines of the cyclides (circles $k_1$, $k_2$, $o_1$, $o_2$) are constructed according to [3], and the points of tangency $R_1$ and $R_2$ of the outlines of the cyclides are found. The cyclide $\Delta_1$ can be acquired through inversion of a conical surface at a sphere. The center of inversion has to be located either at point $M_1$ or at point $M_2$, let us select the point $M_1$. This is the location every infinitely distant point of the conical surface generatrices will be moved. Then the other point ($M_2$) is the location where the conical surface vertex will be transformed. In order to find the vertex $M_2$ of a fitting cone, it is sufficient to perform inversion of $M_2$ (Figure 3). Since the sphere and the cyclide have a common frontal plane of symmetry, it is sufficient to perform inversion in the plane of symmetry at the outline circle $w$. The points of intersection between the outlines of the sphere and the cyclide (between the circle $w$ and $k_1$, $k_2$) remain stationary in transformation and define outline generatrices of the conical surface. Finally, inversion of points $R_1$ and $R_2$ results in points $R_1'$ and $R_2'$ defining the base of the cone. Inversion of its side surface results in the sought part of the cyclide.

$$Radw = 60$$

$$trw = trinv ( ccx (M1, Radw) )$$

$$[Mw2, Rw1, Rw2] = trobj (trw, M2, R1, R2)$$

Frame formation takes the following form:
The cyclide $A_2$ can be acquired through inversion of a cylindrical surface. In this case, the infinitely distant vertex of the cylindrical surface is moved into point $H_0$. In order to do that, the sphere has to be centered at $H_0$. Sphere radius can be arbitrary; the axis of the cylindrical surface has to be perpendicular to $H_0J_1$. As in the first case, in order to perform construction, it is sufficient to consider planar inversion at the outline $g$ of the sphere. Let us find the images $R_{g1}^i$ and $R_{g2}^i$ of points $R_1$ and $R_2$ and construct outline generatrices of a cylinder through them (Figure 3(a)). Inversion of the cylindrical half-surface bounded by a base passing through $R_{g1}^i$ and $R_{g2}^i$ results in the sought part of the cyclide. Since the cyclide thins out approaching to point $H_0$, it is more practical to apply a cylinder bounded on both sides.

\[
O_{fsg} = 1000
\]

\[
trg = \text{trinv}( \, \text{ccx}(H0, psp(o1,0),) \, )
\]

\([Rg1, Rg2] = \text{trobj}(trg, R1, R2)\]

\([Eg1, Eg2] = \text{translate}(0, O_{fsg}; Rg1, Rg2)\]

\[\text{for (12) Pg1 in (Rg1: Eg1], Pg2 in (Rg2: Eg2]}\]

\[\text{for (22) Pw1 in (Mw1:Rw1], Pw2 in (Mw2:Rw2]}\]

\[ [P1,P2] = \text{trobj}(trw, Pw1, Pw2)\]

\[d0 = \text{sab}(P1,P2)\]

\[[s0,_] = \text{c2p}(P1,P2)\]

\[!\text{cout3d}(\, \text{prow} (\text{circz}, s0:)\, )\]

\[^{\mathrm{end}}\text{for}\]

\[^{\mathrm{end}}\text{for}\]

The solution did not require the application of spatial inversion, since formation of the point frame of the circle defined in projection as a segment equal to diameter was previously realized through construction $\text{circz}[3]$ that was used here as well (Figures 3 (b) – (c)).

Example 2. Let us realize construction of a point in spatial inversion transformation illustrated on Figure 4 (a). Let the sphere determining the transformation be defined in the form of outlines $c_1$ and $c_2$. 

---

Figure 3. Construction of a dome element model via inversion.
(\(O_1\) and \(O_2\) are projections of sphere center), while the transformed point is defined by projections \(P_1\) and \(P_2\). Let us construct a triangle \(O_1P_1P_s\) where \(P_1P_s\) equals \(\Delta Z\). Now \(O_1P_1\) equals the distance between \(O\) and \(P\) in space. Let us perform the inversion at \(c_1\): \(P_s\) is transformed into \(P_i\). Let us construct the projections \(P_{i1}\) and \(P_{i2}\).

\[
t = \text{trinv} (c_1)
\]

\[
\text{sub inv3d}(P1, P2)
\]

\[
sl = \text{sab}(O1, P1)
\]

\[
s2 = \text{sab}(O2, P2)
\]

\[
r1 = \text{soa}(s1, P1)
\]

\[
Ps = \text{pab}(r1, rd (s1, mdy(P2, O2)))
\]

\[
Pi = \text{trobj}(t, Ps)
\]

\[
Pi1 = \text{pab}(spa(r1, Pi), s1)
\]

\[
Pi2 = \text{pab}(\text{vr}(Pi1), s2)
\]

\[
!\text{erase}(s1, s2, r1, Ps, Pi)
\]

\[
\text{ret}(Pi1, Pi2)
\]

\end sub

\[\]

Figure 4. Modeling of a spatial inversion.

At this point, it is possible to perform inversion of any three-dimensional object by providing automatic iteration over its points and formation of pairs of projections. Figure 4 (b) depicts inversion of an elliptic torus resulting in owl-like surfaces very similar to surfaces acquired earlier in [5] via the new geometric correspondence called quasi-rotation. Figure 4 (c) depicts another surface acquired through the same program, but with a different location and parameters of the torus. The capability to provide readily available new variants of surfaces via variation of the parameters of the initial figures enables the search for new geometric shapes for architecture and design.

5. Consideration of the results
The usage of the geometry construction language significantly facilitates modeling of curves and surfaces acquired through a certain geometric correspondence or transformation, if the result can be
specified via construction algorithm. The capability to realize new correspondences and transformations, that is to use them as instruments in modeling, makes it possible not only to solve the tasks of creating models, but also to experiment and discover new geometric shapes. In addition, the inv3d construction specified earlier can be applied to perform spatial inversion in solution to any other problem. All that is required is to define projections of the center \( O_1, O_2 \) and to create a transformation object \( t \).

Conclusions
To summarize the results of the study, the aim is achieved. Based on the analysis of geometric transformations theory, new elements extending the GC language have been proposed. These new elements allow one to create models of geometric transformations and to apply these models in solutions to various problems.

The geometric transformations and correspondences can find application in solutions to the problems of modeling in engineering [6, 7], architecture [4], geometric fractals [8], design, etc. Any geometric construction or correspondence that can be defined constructively can be implemented through the GC language.

The author has not encountered previous studies that would propose to implement geometric transformations in the form of independent objects of a geometric model.

Prospects of the proposed method include:
- extension of the existing set of built-in commands, in particular, to enable operations on generalized figures (splines), for example, through methods considered in [9,10];
- implementation of the GC language interpreter into a CAD system.

References
[1] Gans D 1969 Transformations and geometries (New York, Appleton-Century-Crofts) p 424
[2] Peklich V A 2000 Higher descriptive geometry (Moscow: ASV Publ.) p 344
[3] Boykov A A 2021 Development and application of the geometry constructions language to building computer geometric models J. of Phys.: Conf. Ser. 1901 012058
[4] Dukanovic G, Dordevic D, Obradovic M D and Misic S 2014 Application of curves and surfaces of higher orders obtained by inversion in the practice of architecture Proc. 16th Int. conf. on geometry and graphics (Austria: Innsbruck) pp 45-53
[5] Beglov I A 2021 Mass-centering characteristics of solids within quasi-rotation surfaces J. of Phys.: Conf. Ser. 1791 012035
[6] Panchuk K L et al 2017 The mathematical and computer modeling of the worm tool shaping J. Phys.: Conf. Ser. 858 012024
[7] Kokareva Y A 2017 Construction of Canal Surfaces With Variable Generatrix And Parallelism Plane Based on Plane’s Equiaffine Transformations Geometry and Graphics 1 pp 12-20
[8] Zhikharev L A 2021 A Sierpiński triangle geometric algorithm for generating stronger structures J. of Phys.: Conf. Ser. 1901 012066
[9] Karpov D A and Struchenkov V I 2019 Dynamic programming as a method of spline approximation in the CAD Systems of Linear Constructions Russian Technological J. 7 3 pp 77-88
[10] Konopatskiy E V 2021 Principles of solid modelling in point calculus J. Phys.: Conf. Ser. 1901 012063

Source of Funding, Acknowledgements
The work was carried out within the framework of scientific topic No. 140-IRTS RTU MIREA.

The author thanks A V Ermakov for his assistance in making the English translation of the paper.