Institute of Information Technologies
Russian Research Center "Kurchatov Institute"
and
Semantic Segmentation Laboratory

Semantic Segmentation Technology

Project 1

Objects extraction and occlusion relations in a scene from its single color image

Investigators:
A. Akhriev,
A. Bonch-Osmolovsky,
A. Prusakov

Moscow 2000

1. Problem Statement

A major objective of the first stage of the Semantic Segmentation Project 1 is development of model-independent methods of producing an *occlusion graph* that reflects occlusion relations between objects in a scene from its single color image and results of image segmentation into uniformly colored (and possibly textured) regions. The possibility of deriving the occlusion graph is based on the universal occlusion semantics. This term is introduced in the project requirements specification and explained as follows. «The basic visual semantics of any object consists in the fact that it can be occluded by other objects and can itself be an occluder… Occluding (bounding) contours can form junctions and enter into other relations which are to be investigated under this project and used for semantic segmentation. Its final goal is to represent an image as a superposition of a finite number of objects interrelated to each other by the graph of occlusion».

The occlusion graph is important for at least three reasons:

(i) it captures the geometry of the spatial arrangement of objects in the scene in the most general qualitative form when no spatial quantitative data can, in principle, be obtained (note that we deal with a single image of a scene composed of several, at this point unknown, objects);

(ii) it constitutes an important step in handling the image oversegmentation problem (which may totally prevent or make difficult object recognition);

(iii) it can greatly reduce the computational cost of recognizing objects in the scene by first directing the search to objects fully visible in the image (i. e. not occluded) and, therefore, having a good chance to be recognized by straightforward matching against the given database. Upon recognition of these first-level objects, we can proceed to second-level objects (occluded by first-level objects) and so forth.

The analysis shows that constructing a single occlusion graph consistently describing the entire scene may not be possible under some circumstances and conditions. In some cases, the problem is inherently ambiguous and has multiple solutions that cannot be eliminated unless more detailed and accurate input data are given or additional constraints are introduced. Notice that every node of the occlusion graph may correspond to several regions in the presegmented image, if we have reasons to assume that those regions are merely parts of the same surface of one object (possibly differently colored or shadowed) and are simply connected and not separated by an occlusion boundary. In a sense, all we have to know is *what edges are not occluding boundaries*! In general, the more restrictive the constraints on the observed visual world are and the more data is available, the fewer interpretations of the image are possible. All these issues are to be mathematically and computationally analyzed within this project.

2. Defining and Simulating the Input Data

2.0. Assumptions relating to input data generation

Our analysis will be based on the original image and a presegmented image of a scene. The original image will be of use at later segmentation stages and so we focus now on what is to be expected of the presegmented image. In the sequel, unless explicitly stated, by a region will mean a region in a presegmented image. We suppose that the entire original image is split into uniformly colored regions and that such segmentation will not be affected by reasonable amount of object shading. In other words we expect to have as input a pseudo-color image with all its regions set in different pseudo-colors that correspond to different *connected* uniformly colored regions in the original image. Even if two unconnected regions have the same color, they are labeled by different pseudo-colors. Also, two regions having the same color but notably different intensities (brightness) and not linked by a region of gradual intensity change are flooded with different pseudo-colors.
Thus the existence of a border between any two adjacent regions in the presegmented image is a direct indication that one or several of the following visual events have occurred in the original scene:

- an abrupt change in depth (a case of partial occlusion or self-occlusion); the surface normal of the occluding object can (a) either gradually turn away from the viewer as the bounding contour is approached (such a contour is sometimes called a limb), or (b) preserve its orientation in the neighborhood of the bounding contour (an edge-like occlusion);
- an abrupt change in surface orientation associated with a surface edge;
- an abrupt change in surface color resulting from some natural or human-controlled process;
- an edge of a shadow cast by one object over another object or the background.

We assume that the borders between flooded regions are sufficiently well delineated and we know all points where three or more regions meet (junction points or nodes). Moreover, we assume that the angles made by region borders at such junctions are known accurately enough to allow junction classification according to the scheme described in section 5. In the image, there can be abrupt changes in the direction of the borderline tangent (such a point is called a border node, or an *L*-node) and we assume that all such locations are more or less accurately known. Let us also assume that border segments between junctions and *L*-nodes can be accurately enough approximated by straight lines or splines.

Let us finally assume that the preliminary intensity and color analysis carried out in the neighborhood of the borderline *can sometimes limit the list of its interpretations to one or several types listed above.*

These assumptions are only a starting point for the investigation. As we shall see later, if we are to deal with the world of both convex and concave smooth objects, we may have to introduce and analyze unclosed occluding contours (due to partial self-occlusion). And such contours will not necessarily be boundaries of closed uniformly colored regions, and this makes a point for further discussion.

2.1. Procedure and software for input data simulation

The procedure for manual presegmentation of an image consists of several steps. First, we use Adobe Photoshop to manually isolate and paint connected regions of uniform color in one pseudo-color. The main criteria for region isolation are listed in sect. 2.0; and this relates to the region-growing step. In image segmentation, we mostly rely on edges that can be automaticly extracted in Photoshop. These are sometimes corrected in line with our understanding of how well a dedicated color segmentation procedure could work.

The next step is done using a WINDOWS application written in MS Visual C++ 5.0. which allows regions to be counted and related to their counterparts in the original image; boundaries between regions drawn and labeled with the initial set of labels; and their junction types specified. This application consists of two different parts: the interface part and the «computation engine». The interface part allows the user to

- load the presegmented image, load and save all auxilary files;
- run some «house-keeping» operations like counting regions and their areas, testing the validity of the input data, information on the current data status etc;

- manually or semi-automaticly define junctions in the image and generate region boundaries as straight lines, piece-wise straight lines and splines (to be implemented at a later stage);
- define (actually limit) the set of possible labels that can be assigned to each boundary (see above);
- formulate junction labeling rules (i.e, how a junction of the given type can in principle be labelled).

This application was more difficult to develop than initially expected because provisions had to be made for easy editing of junctions and boundaries, for introduction of labels of new type without having to restart all work from scratch, for internal consistency checks, etc. The results of manual segmentation are stored in several auxiliary files to be used in derivation of an occlusion graph.

Details of junction type determination, boundary labeling and the functionality of the «computation engine» will be discussed later.

3. Major Assumptions

3.0. Illumination

We shall assume that the illumination of the scene is typical for daylight or indoor conditions. In other words, we assume that (a) all visible surfaces are indeed illuminated (i.e. are not entirely black), (b) curved surfaces show some degree of shading, and (c) there are distinct shadows in the scene which *at least in some cases* can be identified as such (by a change in intensity alone over a shadow boundary rather than in color). Such illumination conditions are generally created by a distributed light source (like that produced by a sky or a white ceiling in a room) and several point sources such that for most surface patches there are directions along which the light flux is notably different from its angle average. This type of illumination is expected to give rise to clear-cut shading and shadows. The method we develop in this work does not explicitely require knowledge of the illumination spectrum. To limit the search space, however, it is important that, at least in some cases, we could assume that this or that intensity change corresponds to a shadow, or an object limb.

3.1. Image registration

We shall initially assume that camera resolution (in the spatial and color domains) is sufficiently high to allow accurate measurements of any required parameter. Later we shall analyze how degradation in camera resolution may affect the obtained results («graceful degradation» is to be expected).

3.2. Surface coloration

We shall assume that surface coloration does not *simulate* shading. And, therefore, all changes in image lightness alone (with color remaining constant) are due solely to variations of surface patches with respect to the illuminant.

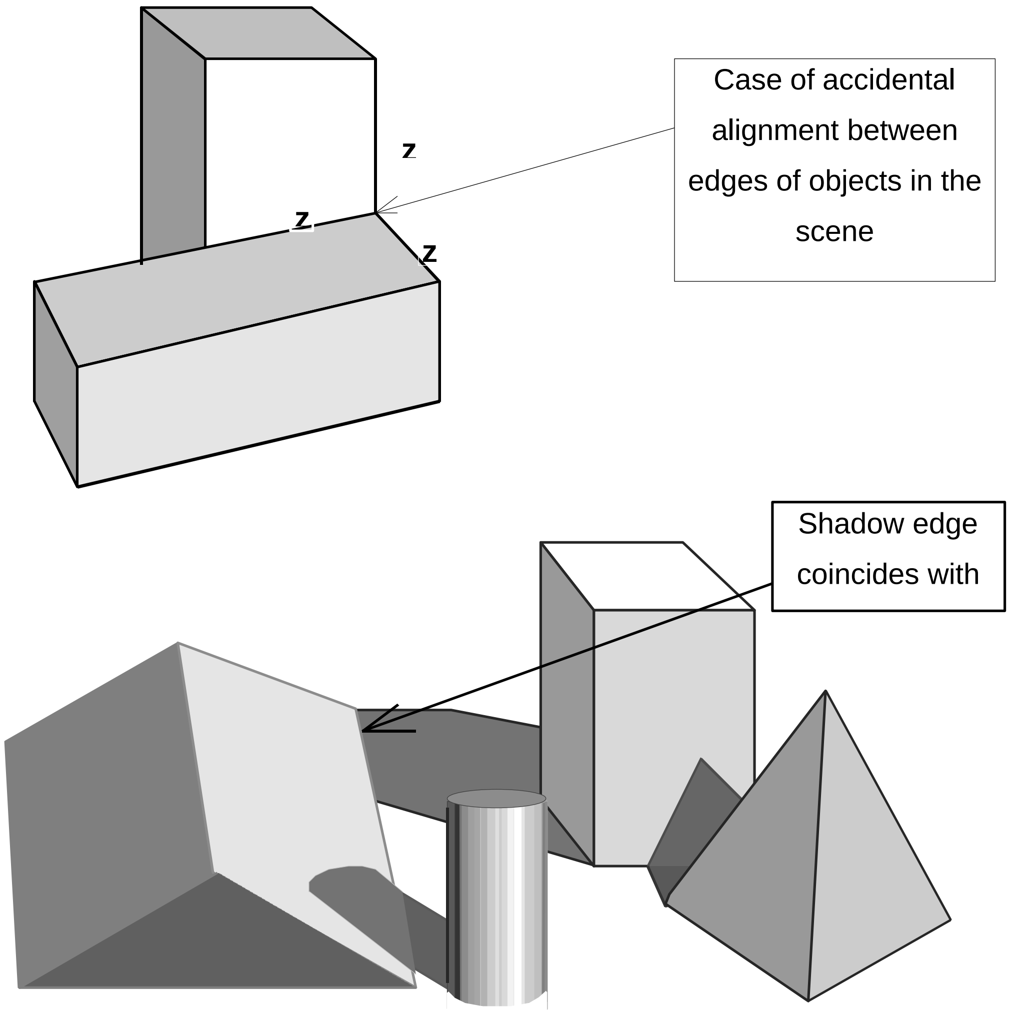
3.3. Consequences

It follows from the outlined assumptions that

- the bounding contour of any object is always visible in the image unless it is occluded by another object of extends behind the frame box;
- nonplane surface regions have lightness and color variations that can be interpreted as resulting from surface warping;
- all unoccluded ridge-edges (lines of abrupt change of surface normal) will be visible in the image;
- all locations of abrupt change of depth in the scene will give rise to discernible changes in lightness and/or color in the image.

3.4 Generic position

This assumption is quite common in computer vision studies. Apart from the object color and/or albedo, the image of a scene is generally determined by the three factors: the geometric arrangement of objects in the scene, the position of the viewer, and the position of the light source. To put it briefly, we shall assume that there is no coincidental alignment between any directions associated with these factors. For example, in the polyhedron world, it means that no plane generated by edges of polyhedra is orthogonal to the image plane (an orthographic projection is assumed) [2]. Also, if a set of projectively-invariant relations between object contours is considered, then small variations of the vantage point and/or the light source position shall not affect the occurrence and type of image events satisfying these relations. Thus, is two straignt lines in the image are parallel, then it can be assumed that, in the scene, they correspond to 3D parallel lines. This because 3D nonparallel lines are mapped into 2D parallel ones only for a zero-measure set of projection directions [3]. A few more subtle examples of how the *generic position* assumprion can be violated are given below in Figs. 2 and 3.



(a)

(b)

Fig.1. Possible violations of the generic position assumption: (a) and (b).

4. Related Work

In our literature survey, we failed to find papers that addressed the model-free semantic segmentation problem precisely as formulated in section 1. It should be once again stressed that what we are interested in at this stage is *not* 3D shape evaluation from wireframes or 2D contour data. We merely want to obtain an occlusion graph that would eventually help in object recognition. There are, however, numerous papers which attempt to recover the 3D shape and locations of objects from their 2D contours of different type using some constraints on their expected shapes and their space arrangement . These works are quite interesting in the sense that they offer a variety of approches that can be partly used in our research.

Pioneering research in this area dates back to Huffman, Clowes and Waltz. In the 1970s, they considered a polyhedral world (scenes composed of polyhedra) and formulated constraints on junction and edge geometry which can be used to find consistent overall scene labelings. Some constraints are more naturally formulated in the so-called *gradient space* (*p, q*) of object face orientations. The solution is obtained by search, or by solving a system of simultaneous equations, or by

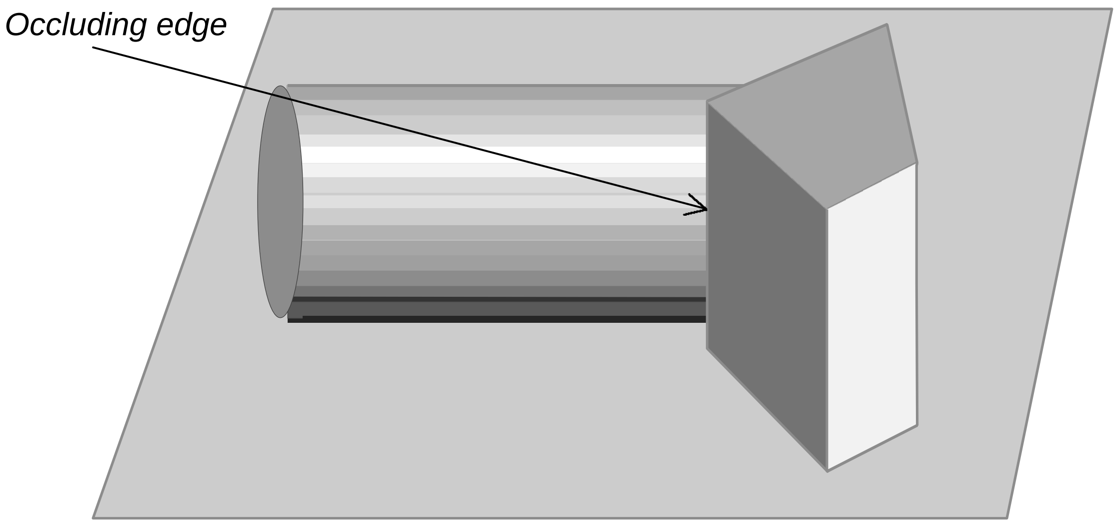


Fig. 2. The generic position assumption can be quite powerful and helpful: suppose that, from the analysis of shading, we know that the surface of the «cylinder» is curved. Then the edge of a prism is an occluding edge and not merely a color region because one should not expect straight lines to arise from lines drawn on a warped surface.

constraint propagation methods, and by other methods [2]. An excelent review of this work and related problems can be found in [4]. Note that some of these constraints originate from topological considerations (such as legal labelings of T-junctions), while others have to do with the specific domain assumptions (planarity of object faces, solid objects, maximum three faces meeting at each junction, etc.).

Kirousis [3] analyzed the problem of effective labeling of planar projections of polyhedra and showded that even in this simple case the labelings problem is NP-complete. He also showed that if additional information was known about edges coming into Y-vertices of polyhedra, the problem could be solved in linear time.

We should also mention ref. [11] in which an attempt is made to segment a scene into objects by using a set of 3D primitives and a matching procedure that works simultaneously at several hierarchical levels: that of edges, aspects, and 3D solids.

In full, our literarure survey will be included in the final stage report.

5. Labeling Schemes

5.0 Label set

In this work, we shall use different labels for occluding edges, edges arising from breaks in surface orientation, color edges, and shadow lines. The range of labels that can, in principle, be assigned to presegmented contours in an image depends of the type of the scene under consideration (e.g., polyhedra, «origami» world, smooth convex solids, smooth convex and concave solids, all these scene types with or without shadows, etc.). Our notation for such labels is similar to that generally accepted, but relies more on letters than on arithmetic signs (+,–, <, >, etc.), which can also be used when there is no chance of confusion.

A contour between two differently colored regions (which also implies the same color but different lightness) in the original image belonging to the same surface will be denoted by **c.**

An edge resulting from a break in surface orientation will be denoted by **e+**, if the two surfaces meeting at this edge form a convex 3D-angle, and by **e-** elsewhere.

A shadow edge will be denoted by **s**. To indicate that the shadow is to the right of the shadow line, we shall write **s<**, and **s>** in the opposite case.
The occlusion line between two objects (a step-wise change in depth) will be denoted by **z**. If the occluder is to the right of this line, the line is labeled by **<,** and by **>**, in the opposite case. Occluding contours corresponding to a limb, (the surface normal and the viewing direction making a right angle at the limb (see sect. 1)), will be denoted by $\mathbf{z_o}$. Notice that points belonging to a limb are in no way different from other surface points. Occluding contours formed by an object edge so that only one object face is visible in the image will be denoted by $\mathbf{z_e}$.

5.1. Junctions and their labeling

By labelling a junction we mean assigning labels to its incoming boundaries. It is well known that only a limited set of labeling of the given junction [4]. For example, in the classic Waltz polyhedral world without shadows and cracks, there are only 4 junctions and 16 labelings. The types of junctions that can occur in the image and

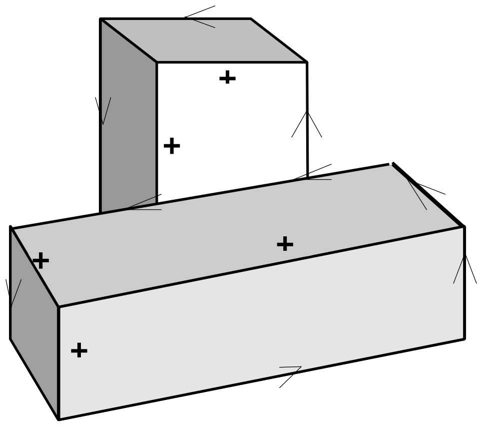


Fig. 3 Example of correct labeling of a polyhedral scene. The corresponding occlusion graph consists of two nodes.

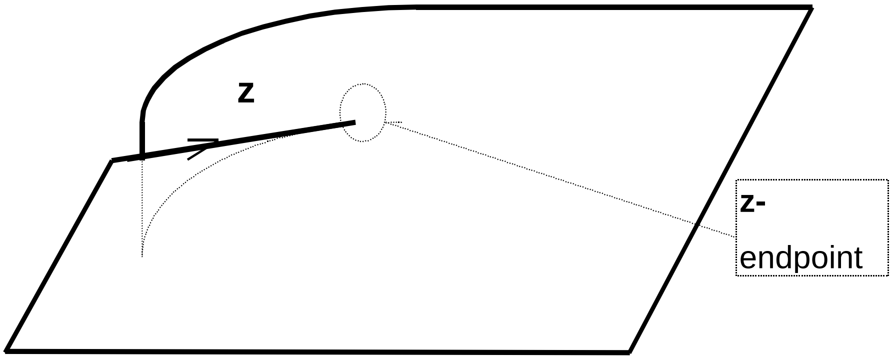
their labelings are determined by the nature of objects making up a scene (polyhedra, smooth convex, piece-wise smooth convex with simple edges, etc.) and by their geometric arrangement. Some junctions and labelings are rulled out by the Generic Position assumption. There are two ways in which the variety of junctions and labelings considered for the given scene can be constrained. One can adopt some more or less general geometric assumptions (like those mentioned above) and derive logically the list of possible junctions. Or one can simply assume that such and such «queer» junctions never appear in the scene. To be able to do that, one has to know what junctions are in principal possible. Accordingly, we shall now discuss the types of junctions that can arise under very general conditions, and formulate what junction-related evens shall be assumed impossible. For logical coherence, we start with 1-junctions (one incoming borderline) and then proceed to higher order junctions.

5.1.1. Endpoints (1-junctions)

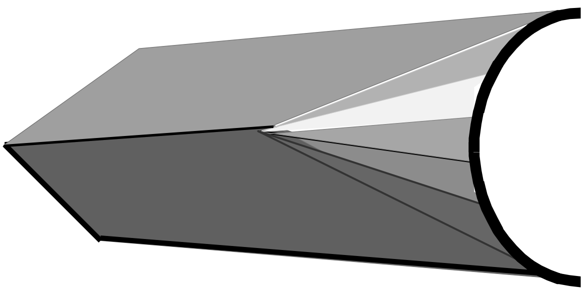
Strictly speaking, such junctions are not entirely legal under this project because the borderlines we consider are assumed to be boundaries of 2D-regions. And, nevertheless, such junctions have to be analyzed because they can occur under very general conditions and, in many cases, are essential for consistent scene labeling. The computational procedure that can generate such boundaries is to be discussed.

(a) Endpoints of a **z**-boundary. For a (piecewise) smooth nonconvex object, there are viewing directions such that one region of the object surface will be occluded by its another part forming something like a fold. As an example consider the internal part of a jug partly obscured by its top rim, or a corrugated and at the same time warped surface. This type of occlusion of will give rise to a segment of a **z**-boundary ending in two endpoints with unstable positions in the sense that small variations of the viewing direction will preserve such points in the image but affect the corresponding points on the object surface.

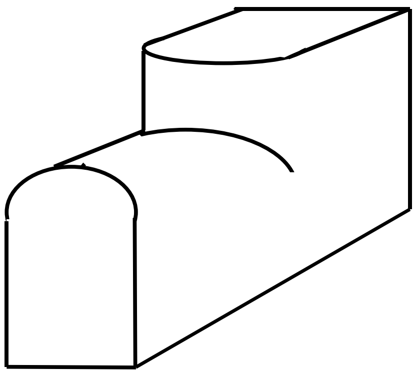
A **z**-boundary endpoint can also occur at places of surface rupture as in the following Figure. Objects of this type will be excluded from our consideration in this work (*Assumption 2*).



(b) Endpoints of an **e**-boundary. Such endpoints can occur in the image of an object having two adjacent plane faces that smoothly change into one curved surface, for example that of a cone, as in the Figure below. Here the **e**-endpoint is the apex of a semicone is, therefore, a point of a none-zero total curvature. In its neighborhood,



there are points with infinitely large mean curvature. This situation is hard to deal with in the context of uniform color segmentation and we shall assume that our objects have no smoothly built-in cones (*Assumption 3*). Here is another example of an **e**-endpoint but in the case of curved faces.

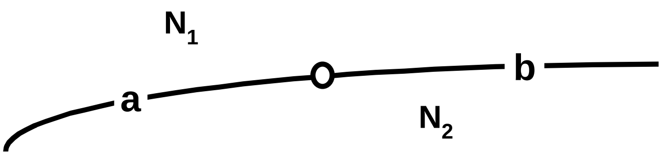


(c) Endpoints of a **c**-boundary. Such an endpoint can arise if the color and lightness in two adjacent regions change along a **c**-boundary such that gradually the difference between them vanishes. In the sequel, we shall assume that *no such events occur in images under consideration.* This is a constraint on admissible object coloration.

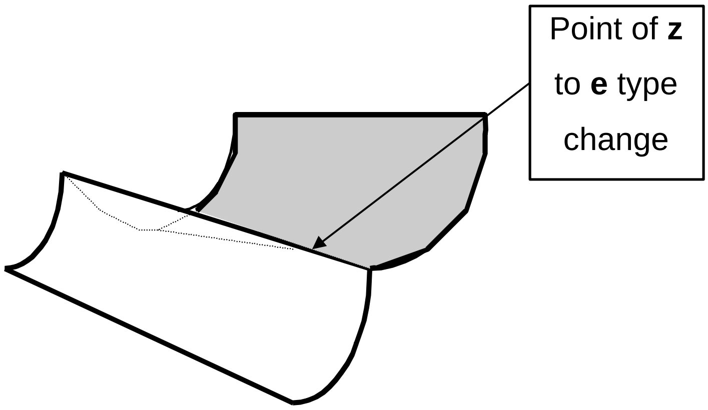
(d) Endpoints of an **s**-boundary. Such points can occur in the image as a result of "smearing" and disappearance of a shadow line due to the illumination conditions in the scene. This never happens where there is a predominant point light source. With increasing share of diffuse lighting, a shadow edge may well become less clear-cut and any edge detection is bound to find its endpoint. We shall assume that the intensity and color analysis along the boundary and in the neighborhood of such a point (by measuring the degree of smearing, which must increase) will allow us to distinguish an **s**-boundary endpoint from a **z**-boundary endpoint.

<u>5.1.2. Type transition along the boundary</u>

Consider the following Figure, where $\mathbf{N_1}$ and $\mathbf{N_2}$ are two regions and **a** and **b** two smoothly adjoining portions of their boundary having different types. This kind of transition is impossible in the Waltz world with plane faces and in the world of smooth convex objects but can well occur under less restrictive conditions.



Here is an example of a **z** to **e** type changeover along a smooth boundary. A part of the surface hides behind the edge. The location of the point where the boundary type changes may vary along the boundary line and is determined by the viewing direction.

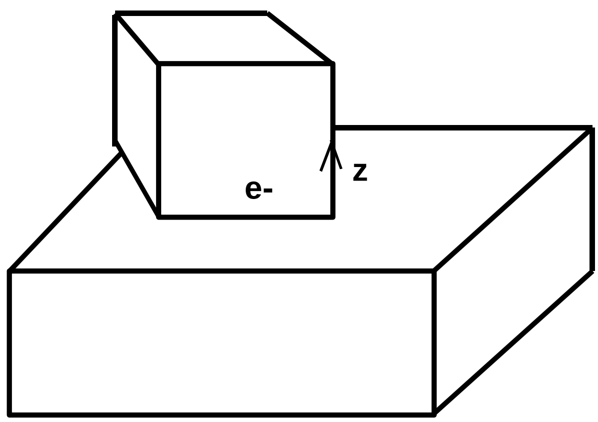


For the time being, we shall exclude from our consideration the situations (objects) where the location of a transition point is determined by the surface geometry alone and is associated with a surface rupture at a fixed point (see subsection 5.1.1.a). For the sake of breivity, we omit discussion of several other cases that can occur in the world of curved Origami objects [3].

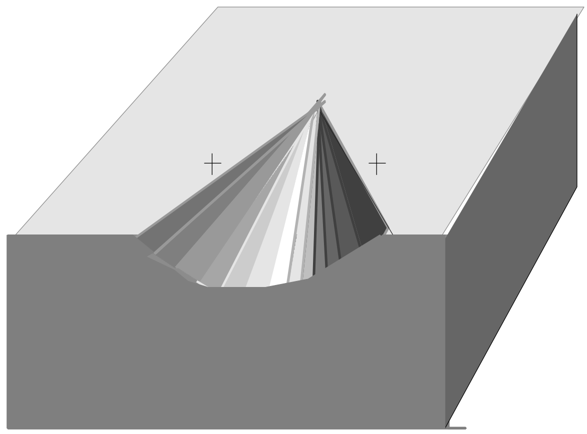
All other boundary type transitions (**z**->**c**, **z**->**s**, **e**->**c**, **e**->**s**, and **c**->**s**) can be exculed from consideration either by the Generic Position assumption or by assumptions already made in this section.

5.1.3. 2-junctions

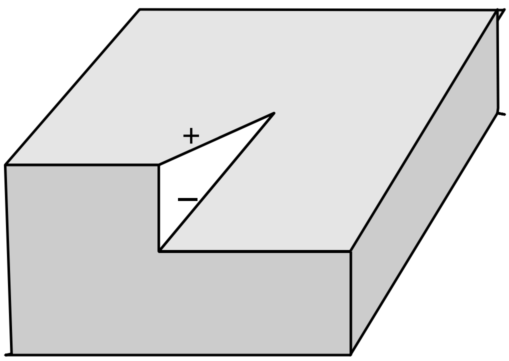
2-junctions (or *L*-nodes) are points were two pieces of boundary make an angle. The two incoming boundaries can have different types even in the polyhedral Waltz world [4], and, in our case, all the transitions between $\mathbf{z_e}$, **e+**, and **e-** types described for this world at *L*-nodes are also possible. An example of a $\mathbf{z_e}$ **->** **e-** transition is given in the following Figure.



One can also think of examples of the **e**->**e** type transitions that are impossible in the polyhedral world. In the following Figure, we have an **e+-> e+** transition, but this object, could be excluded from consideration by Assumption 3.



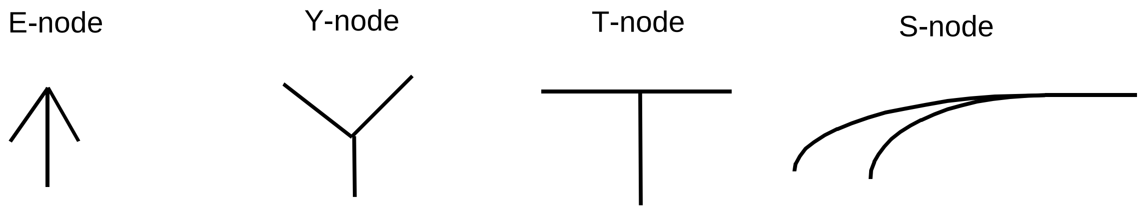
By Assumption 2, we canl also ban the **e+-> e-** type transitions as in the folowing image.



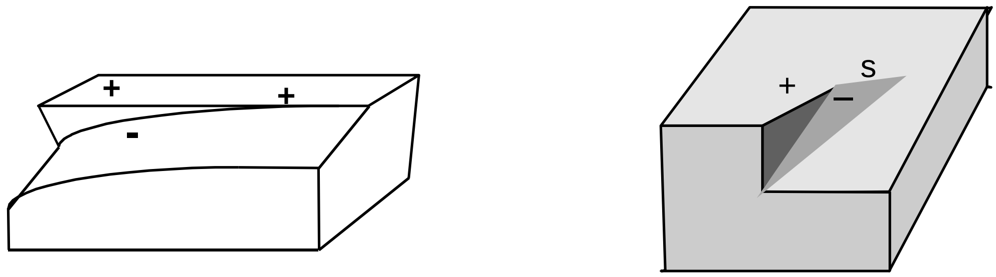
Thus, for 2-junctions, we end up with the conventional labeling variants as in the Waltz world.

<u>5.1.4. 3-junctions</u>

Consider the following four 3-junctions. The first three of them come from the polyhedral world and don't need much explanation, while the fourth one is typical of smooth curved objects. The new S-nodes are important because it can be shown that, under the Generic Position Assumption, any smooth curve on the surface of a smooth object having a common point with the occluding contour will not intersect it but flow into it smoothly, as suggested by the following drawing



In the general case, with no constraints imposed on object shape and color, a wide variety of labelings are possible, as demonstrated by the following examples.



For practical applications, it would be reasonable to introduce some assumptions on the object world we want to deal with and then, for each legal junction type, derive all possible labelings that can be initially assigned to this junction in the image. This is what was done by Waltz for different variants of his blocks world. Otherwise, there is hardly any chance to arrive to a unique solution by constraint propagation method because the constraints will be too weak.

6. Labeling Constraints for Different Worlds

Below we give a list of possible practical constraints on the shape, color and illumination that limit the number of legal junctions and labelings. In the following subsections, we considere such junctions and labelings for several combinations of such constraints that were not previously discussed in literature.

Shape constraints:

- smooth surface (no edges);
- convex objects;
- objects having only three-face vertices (that are intersections of exactly three smooth faces);
- objects with piece-wise smooth faces that have smoothly varying tangent planes up to object edges;
- simply connected objects (no holes);
- solid 3D objects (no hanging faces or edges, no Origami constructuions).

Color-related constraints:

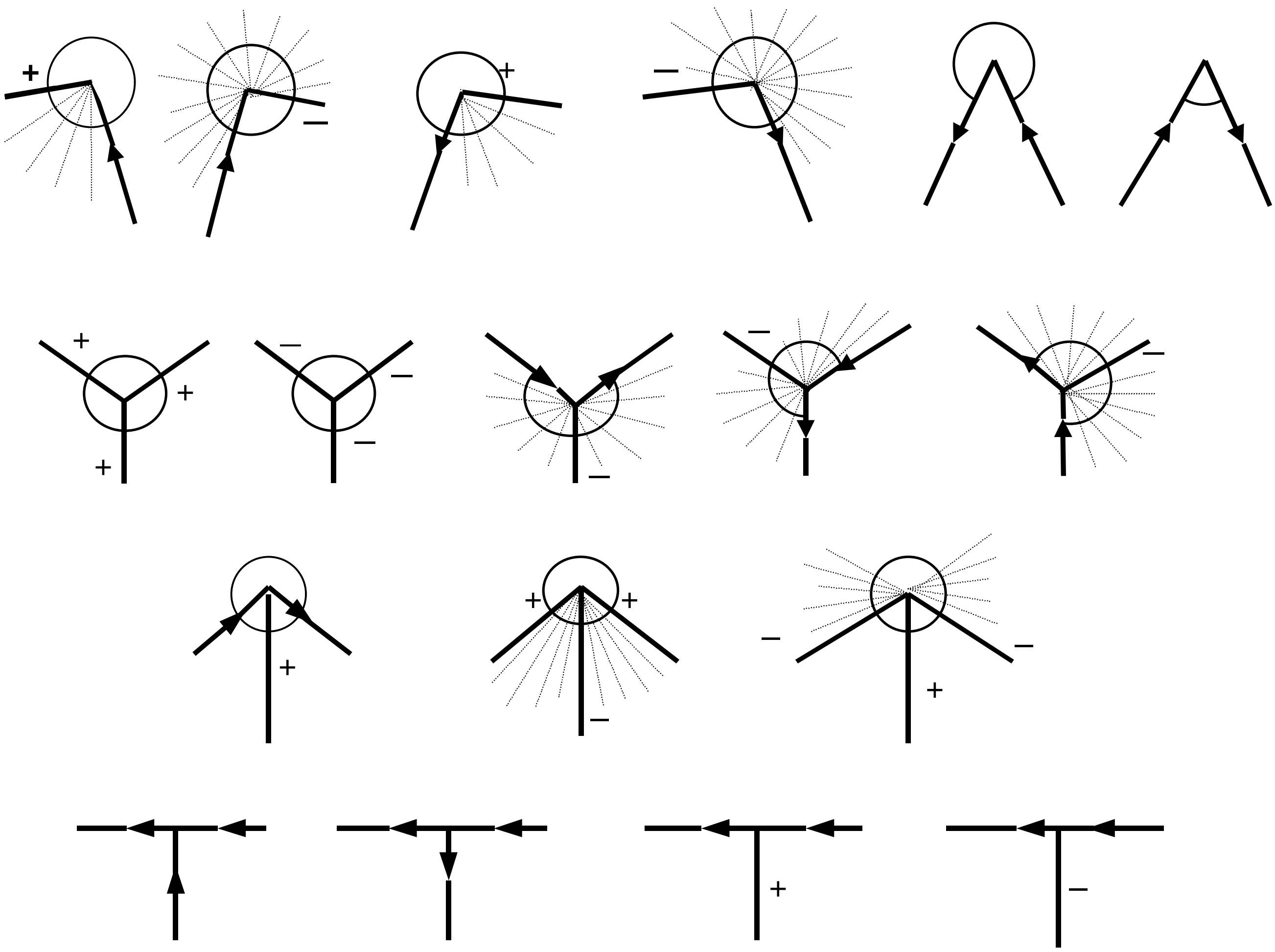
- the entire surface of each object has one surface color;
- color boundaries do not come in polyhedral object vertices (this also applies to vertices with nonplane faces for curved objects).

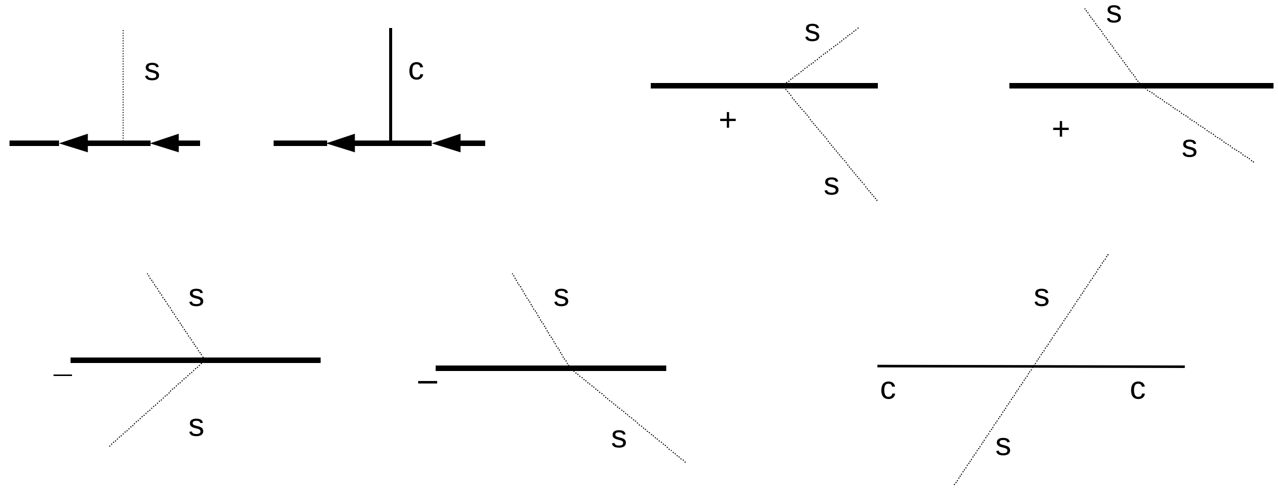
Illumination constraints:

- illumination is such that there is no smearing of shadow lines, that is all shadow edges are clear-cut and have no endpoints;
- the light sources include one strong point source and a diffuse source such that there a predominant illumination direction in the scene.

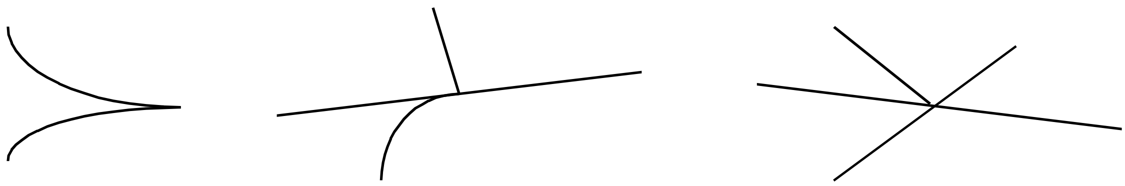
6.1. The Waltz world with shadows and color

Waltz considered a trihedral world of solid objects with similarly colored surfaces [4]. He first described the relations between boundaries in the absence of shadows and then extended his treatment to include shadow boundaries. We can add some new constraints for the same world but having also color surfaces. In our drawings, a fan of broken lines denotes the sector of a node that can possibly contain shadow line(s). The sector that can contain a color boundary coming into the node is denoted by arc of a circle.

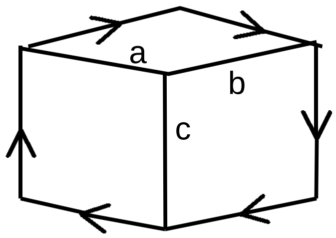




The nodes formed by **c**-boundaries alone can have any form whatever, for example:



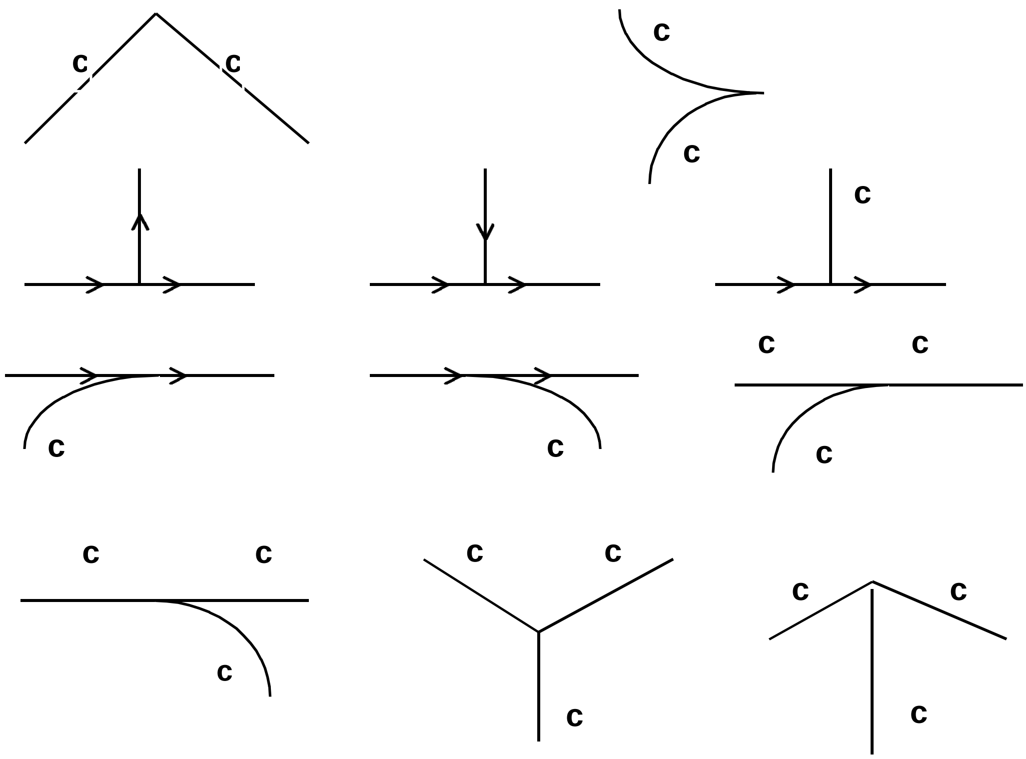
Now let us see if anything has changed in labeling a cube by introduction of color into the Waltz world. Not much, if we adhere to our problem statement (that of calculation of the occlusion graph), because no new hypothesis about occluding contours arises: the a, b, and c edges (see Figure) can now be color boundaries but still cannot correspond to occluding edges.



6.2. Smooth convex colored objects and no shadows

In our analysis, we shall use the fact (see 5.1.4) that among all viewing directions such that a given point on a smooth curve lies on the occluding contour, there is only one direction such that the projection of the curve in the image runs into the occluding contour at a nonzero angle. So, under the General Position Assumption, we can assume that image projections of smooth curves are tangent to the occluding contour. As a result, the following constraints on junctions and their labelings can be formulated for this world:

- only two boundary types are possible in the presegmented image, **z_o and c**;
- no endpoints occur, and there is no changeover of the boundary type along the boundary between two nodes;
- the only nodes that can have **z_o** labels on incoming boundaries are T-nodes and S-nodes. The **c** boundaries can produce any nodes. The legal labelings are listed below.



All other n-junctions, n>3, can be formed only by **c**-boundaries.

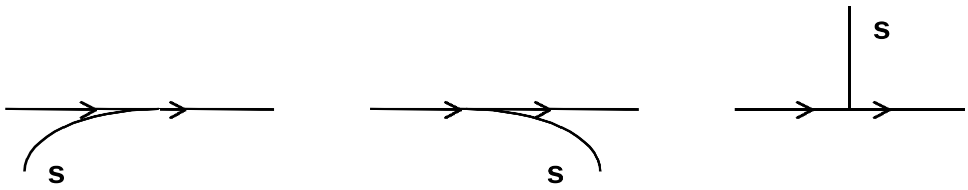
Let us draw some conclusions for the world of smooth convex objects:

- if a group of adjacent regions has a closed boundary consisting of **z_o** elements and no such labels are assigned to internal boundaries within this group, then, together, these regions cover completly the visible surface of some object in the scene;
- any closed boundary of a region group in the presegmented image which has no L-nodes, encloses a convex region, has only S-nodes with incoming tangent lines on the right (for boundary followed in the clockwise direction) and only T-

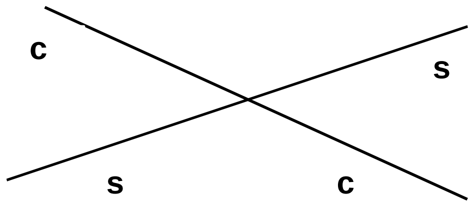
nodes with incoming boundaries on the left can represent a bounding contour of some object in the scene.

6.3. Convex smooth objects with color and shadows

In this case, there will be mainly three border types in the presegmented image, $\mathbf{z_0}$, **c**, and **s**. The **s**-endpoints are possible but do no harm and make the labeling problem even easier by constraining the set of legal labelings at the other node. No boundary type changeover is possible. The set of junctions and labelings includes those valid for smooth convex colored objects (see 6.2) and some new combinations involving a shadow boundary. All the relations between shadow edges and occluding contours meeting in 3-junctions will be similar to those between color edges and occluding bounadies, $\mathbf{z_0}$, and are listed below

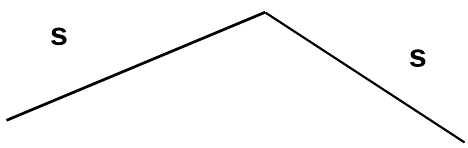


The only node stable under the General Position Assumption is a junction formed by the intersection of a color edge and a shadow edge,



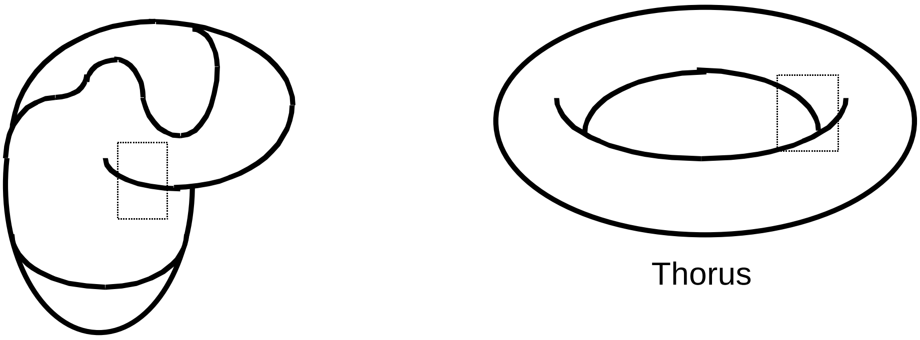
Note that a junction of this type can often be identified by the specific relationship between color and brightness on both sides of the shadow boundary.

Due to one object occluding another with respect to the light source, two shadow boundaries can also meet in an L-node, despite the fact that all objects in the scene are smooth and convex,

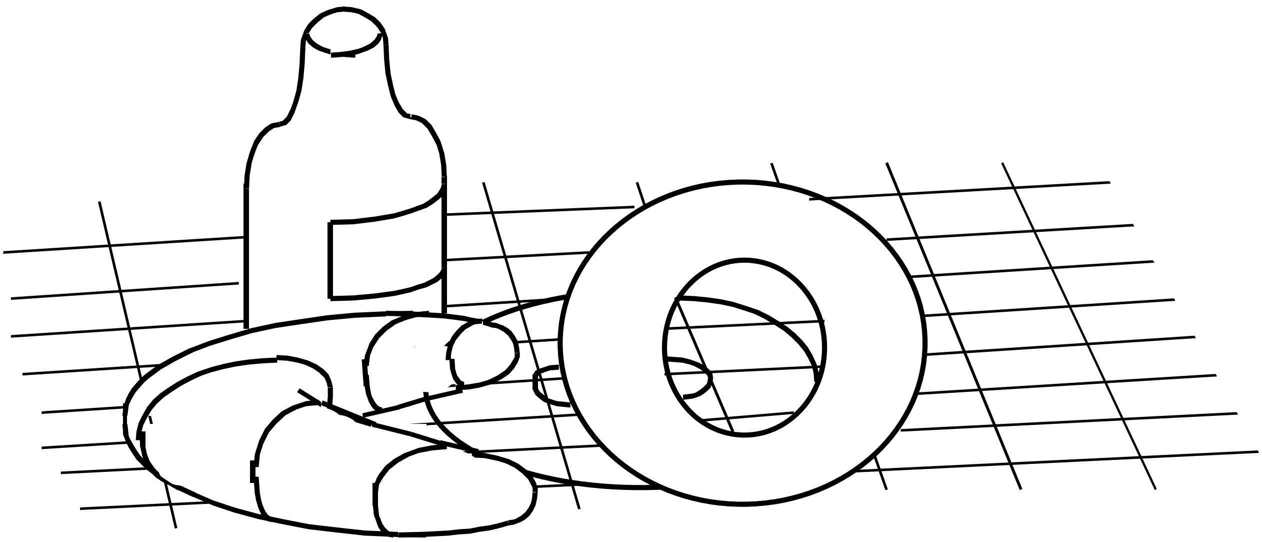


6.4. Nonconvex smooth objects with color and shadows

In this case as well there will be mainly three border types in the presegmented image, $\mathbf{z_0}$, **c**, and **s.** No boundary type changeover is possible. One can expect, however, the occurrence of $\mathbf{z_0}$-endpoints, as discussed in section 5.1.1. Let us stress once again that such boundary segments do not divide closed and connected regions of the presegmented image. But we insist that the segmentation procedure must somehow be able to produce such contour segments at least in the case when they are clearly defined in the image and separate pixels that would be definitely assigned by *local* analysis to different regions. Two examples of images of this kind of objects having a $\mathbf{z_0}$-endpoint is given in the following drawings



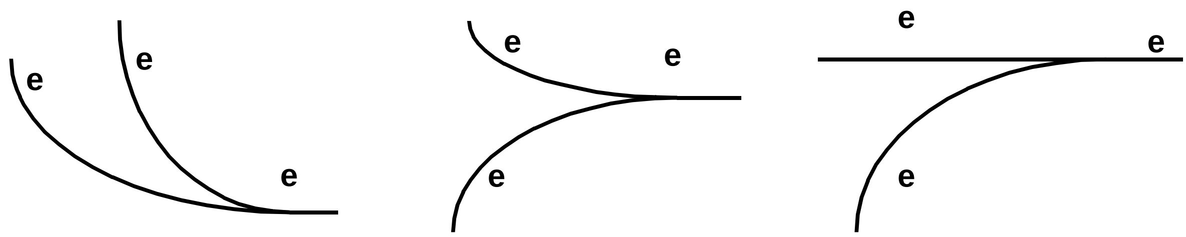
The relations between incoming boundaries for 2- and 3-junctions will be those valid for smooth convex colored objects with shadows (sect. 6.3). It can be shown that the labeling rules and constraints so far formulated are sufficient to arrive to a single labeling solution for the following scene



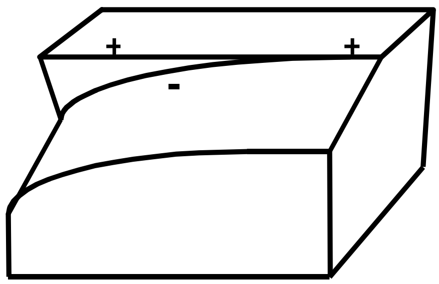
6.5 Solid, piecewise smooth, nonconvex objects with no more than trihedral vertices, colored faces, shadows and no tangential edge splitting (bifurcation)

To clarify the significance of these assumptions, let us consider some boundary type combinations at junctions that become impossible under these constraints.

By banning smooth tangential splitting of surface edges, we exclude situations when boundaries in the presegmented image coming tangentially into any junction arise from surface edges. This also implies that no two edges can meet tangentially at a junction as in the following drawings



This is an example of an object not allowed under the adopted constraints.



In addition, we do not allow a change from the $\mathbf{z_e}$ bondary type to **e+** when such a change occurs along a smooth boundary at a fixed surface point (see 5.1.2).

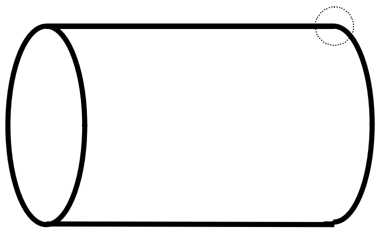
The object faces are assumed to be piecewisely smooth up to and including edges. This implies that every smooth surface patch can be smoothly extended over and beyond the edge line and will have regular and smoothly varying tangent planes in the 3D space at all points of such an extention including those belonging to the edge. This constraint, quite reasonable, for most natural objects, bans the following relations between boundaries meeting at junctions (as usual, under the Generic Position Assumption)

- the **å-** and $\mathbf{z_e}$-endpoints;
- type change along a smooth boundary from **c** to **e**, and from **e+** to **e-**;
- L-nodes with the following labelings (**e+**, **e-**), (**e+**, **e+**), (**e-**, **e-**), (**z**, **s**), (**z**, **c**);

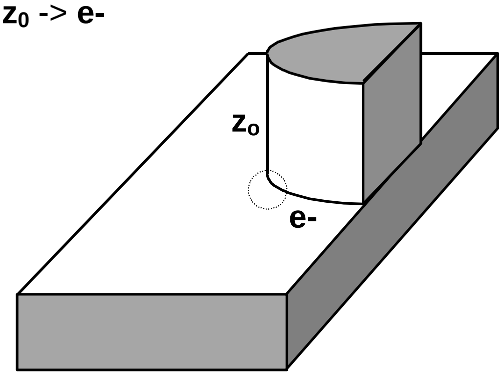
Presegmented images of objects belonging to the world under consideration are allowed to contain all the four types of occluding boundaries (clockwise and anticlockwise directions of a bounding contour $\mathbf{z_o}$ and two directions of the edge occlusion $\mathbf{z_e}$), convex and concave edge contours, **e+** and **e-**, color boundaries **c**, and shadow lines **s**. The $\mathbf{z_o}$- and **s-**endpoints are also possible.

The following boundary type changes are allowed in this world along a smooth boundary:

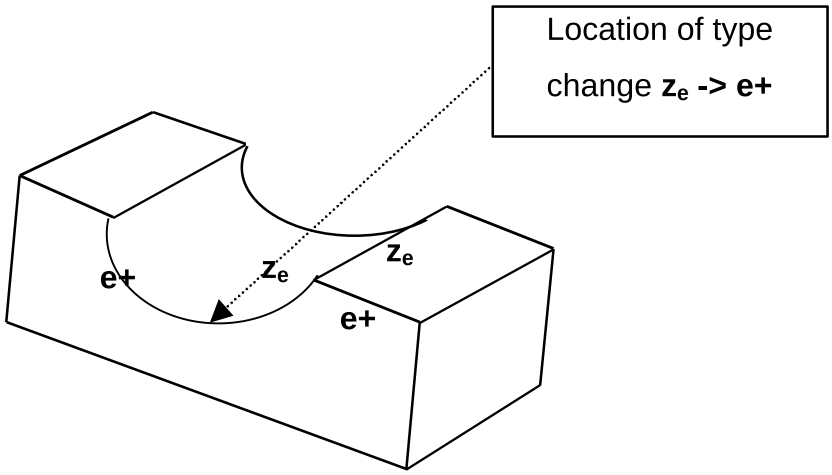
z_0 -> z_e



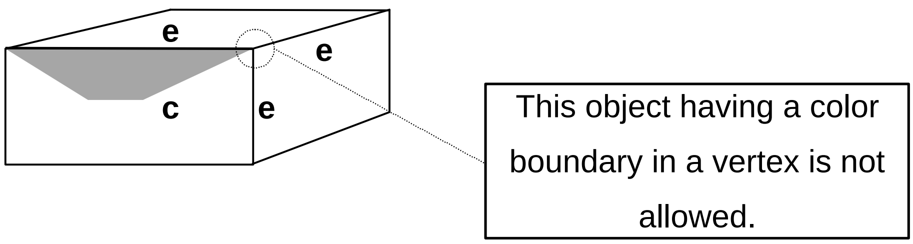
z_0 -> **e-**



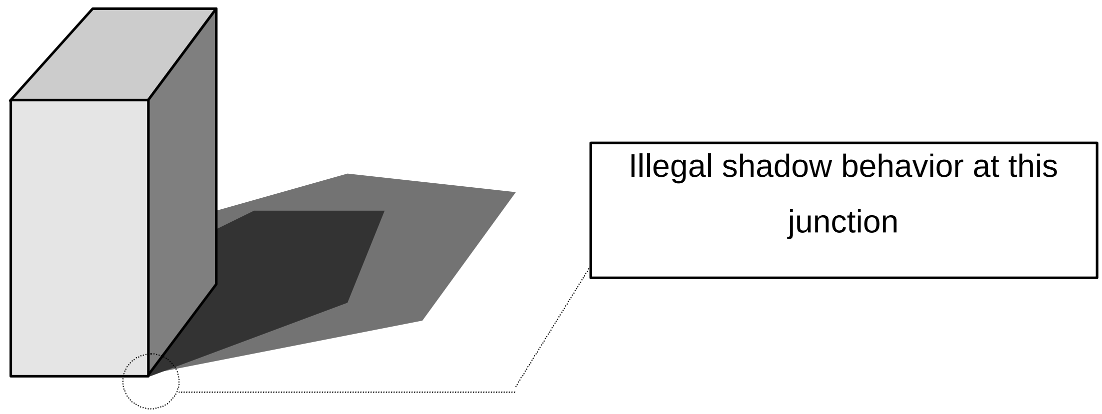
and z_e -> **e+**



Let us finally illustrate two constraints on color and illumination. The color boundaries are not allowed to enter nodes that include edge boundaries.



The requirement of a single predominant source of illumination prohibits two embedded shadow regions as well as two shadow boundaries coming out of one junction.



7. Limiting Initial Boundary Labeling by Intensity Analysis, Parallelism and Skewed Symmetry

The calculation of the occusion graph for the given image of a scene involves three steps:

- identifying region boundaries and their junctions in the image and assigning the initial set of labels to these boundaries;
- search for consistent solutions of the labeling problem for all junctions by some constraint propagation method or a search procedure;
- merging of adjacent region that belong to one object as implied by the obtained labeling and construction of the occlusion graph.

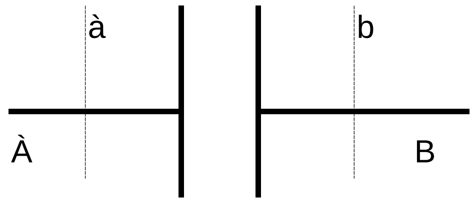
A crucial point is limiting the search space both for computational efficiency and to eliminate some labelable but not realizable solutions [3]. This is very important because, as discussed in sect. 3, the labeling problem is NP-complete and may require exponential times and have an exponential number of solutions.

As an example, consider the world of convex objects with color and no shadows (sect. 6.2) and an image represented by *n* concentric circles of different color. It is obvious that each circle could represent either a color patch on the underlying object (say a sphere) or a separate solid object. Thus, in this case, there are 2^n solutions to the labeling problem and an enormous number of occlusion graphs.

This, however, should not sound too discouraging, given that in many reasonable cases the algorithms previuosly suggested showed good running times. Whenever there is a sufficient number of T-, Y, and S- junctions, the number of solutions is very limited. This is also demonstrated by examples considered in sec.

8. We have attempted mathematical analysis of this problem and arrived to some results relating to the number of solutions possible under different circumstances, but these have still to be elaborated and will be presented later. Anyway, the amount of search could be reduced if we could constrain the list of labels initially assigned to each boundary.

It was already noted by Kanade [3] that, by studying image edge profiles (that is intensity and color variations across some boundary segment), it is sometimes possible to limit the list of valid interpretation for this boundary. Thus a peak-type profile suggests a convex edge, a roof-type a concave edge, a change in intensity but not color a shadow edge, a change in color means definitely a non-shadow boundary, and a step change of color might mean a color boundary or a z_e boundary. Such an absolute approach may be unreliable under certain conditions. A more promising method (termed "relative" by Kanade) consists in comparing edge profiles for boundaries forming certain specific geometrical configurations, like matched T-nodes in the following Figure. If we find that



color edge profiles are "similar" for boundaries A and B, then (i) the labels for A and B are likely to be the same, and (ii) A and B are likely to be projections of a boundary of some object occluded by another object. Similar arguments can by applied to matched ð-configurations (constituted by two successive T-nodes with similar orientations). Kanade suggested some formalization of these concepts (assigning weights to different labelings based on the measure of edge profile likeness) and we currently investigate to what extent these can be used in our work.

Another source of constraints on boundary labeling is the use of parallelism and skewed-symmetry, which are assumed to be of non-accidental nature, and, therefore, imply similar interpretation for the corresponding boundaries. The methods for detection of parallelism and skewed-symmetry are discussed in Refs. [6-10]. We are currently looking into how these methods can be incorporated in our algorithms to trim the search tree.

8. Implementation and Preliminary Results

We have developed and implemented in the C++ language a recursive procedure that finds all the solutions to the boundary labeling problem. This procedure combines the search with operations that effectively limit the search space by testing the coincidence of labelings that can be assigned to adjacent junctions at each step and filtering out nonmatching label combinations. This procedure is incorporated into the "computation engine" of our software application discussed in sect 2.1.

The power of the labeling schemes considered above was tested for different worlds on several simulated images created using 3D Studio software and shown in Appendix 2. These images are in files cx.tif and cx_p.bmp (smooth convex color objects with shadows), ncx.tif and ncx_p.bmp (nonconvex smooth color objects with shadows) and ncxed.tif and ncxed_p.bmp (nonconvex piecewise smooth color objects). The corresponding *.bmp files are presegmented images of the scene.

Let us first consider the case of smooth convex color objects with shadows shown in Fig A1. We assume that region (0) corresponds to the background. Then regions (9) and (10) have to be merged into one surface region because their boundary contains an L-junction (C) which can be produced only by adjacent color regions and/or shadows. All external boundaries of these regions will be occluding contours as indicated by the T-junctions (F), (G), (J), (K), and (E) and S-junctions (I) and (H). It follows then that regions (9) and (10) completely cover the visible side of one object. By similar arguments, it is readily shown that regions (5), (6), (7), and (8) represent the whole visible surface of another object, and that regions (1), (2), (3) and (4) should be merged as different patches of the third object. Note that it is immaterial to us whether the (A) - (J) boundary is a color edge or a shadow edge.

Consider now Fig. A2, representing nonconvex smooth color objects with shadows. Each boundary segment of region (8) can correspond only to occluding boundary, as demostrated by the T-junctions (D), (E), and (F). We can trace this boundary in the clockwise direction and come to the starting point having met only the left-turn T-junctions. So this region corresponds to a single unoccluded object. The point C can correspond only to a color or shadow/color junction, and this allows us to merge regions (4), (5), (6), and (7) into one surface region which occludes regions (2) and (3) (as demonstrated by the T-junctions (G), (H), and (I)). Finally, we should merge regions (1), (2), and (3) because the point A is an L-junction, and this corresponds either to a color edge or a shadow edge, and the point B is an S-junction. So in this case as well we find a single three-object solution.

In Fig. A3, we have an image of a scene containing piecewise smooth nonconvex objects. The reasoning in this case is more complex and cumbersome and we do not report it here. In this case as well there is a single soluition to the labeling pioblem and we can identify three objects in this image. We are now in the process of testing our program on this image.

10. References

1. Semantic Segmentation Technologies, R&D Project and Research Plan, 1999.
2. Kirousis, L.M., Effectively Labeling Planar Projections of Polyhedra, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, pp. 123-130, 1990.
3. Kanade, T., Recovery of the 3-dimensional Shape of an Object from a Single View, *Artificial Intelligence,* vol. 17, pp. 409-460, 1981.
4. Winston, P.H., *Artificial Intelligence*, 2nd ed., Reading, MA, Addison-Wesley, 1984.
5. Kanade,T and Kender, J.R, Mapping Image Properties into Shape Constraints: Skewed Symmetry, Affine-Transformable Patterns, and the Shape from-Texture Paradigm, J. Beck, B. Hope, and A. Rosenfeld (Eds.) *Human and Machine Vision,* Academic Press, N.-Y., pp. 237-257, 1983
6. Terzopoulos, D., Witkin, A., and Kass, M., Symmetry Seeking Models and Object Reconstruction, *International Journal of Computer Vision* vol.1, pp. 211-221, 1987.
7. Nalwa, V.S., Line-drawing Interpretation: Bilateral Symmetry, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no.10, pp. 1117-1120, 1989.
8. Ponce, J., On Characterizing Ribbons and Finding Skewed Symmetries, *CVGIP* vol. 52, pp. 328-340, 1990.
9. Ulupinar, F, and Nevatia, R, Perception of 3-D Surface from 2-D Contours, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no.1, pp. 3-18, 1993.
10. Gross, A.D. and Boult, T.E. Analyzing Skewed Symmetry, *International Journal of Computer Vision,* vol.13, no.1, pp. 91-111, 1994.
11. Dickinson, S.J., Pentland, A.P., and Rosenfeld, A., 3-D Shape Recovery Using Distributed Aspect Matching, *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 14, no. 2, pp. 174-198, 1992.
12. Kanatani, K. Computational Projective Geometry, *CVGIP: Image Understanding,* vol. 54, no. 3, pp. 333-346, 1991.
13. Nielsen, L. and Spar, G. Projective Area-Invariants as an Extension of the Cross-Ratio, *CVGIP: Image Understanding*, vol. 54, no. 1, pp. 145-159, 1991.
14. Pizlo, Z., and Rosenfeld, A. Recognition of Planar Shapes from Perspective Images Using Contour-Based Invariants, *CVGIP: Image Understanding*, vol. 56, no. 3, pp. 330--350, 1992

15. Barrett, E.B., Payton, P.M., Haag, N.N., and Brill, M.H., General Methods for Determining Projective Invariants in Imagery, *CVGIP: Image Understanding*, vol. 53, no. 1, pp. 46-65, 1991.
16. *Geometric Invariance in Computer Vision*, Mundy, J.L, and Zisserman, A, Eds., The MIT Press, Cambridge, Massachusetts, 1992.

Appendix 1. Internal Data Representation

```
/*=================

Coordinates: The left bottom image corner is (0,0). The X-axis
is horizontal. Angles are measured from the X-axis in the
counter clock-wise direction.
All data points are assumed to have unique image coordinates
so that different points have different coordinates.
Data Arrays: The two main dynamic data arrays used are
ArrayDot and ArrayBound. ArrayDot stores pointers to all
junctions described by the DotMain class and differing by
their types (see UINT TypeDot).
ArrayDot is defined as follows:
 =================*/

class BoundDot;  // see below
typedef CTypedPtrArray<CPtrArray, BoundDot*> ArrayBoundDot;

class DotMain
{public:
     UINT TypeDot; // type of junction
     CPoint Dot;   // image coordinates
     ArrayBoundDot* ArrDT; // points linked to this
                           // one by region boundaries
};

typedef CTypedPtrArray<CPtrArray, DotMain*> ArrayDot;
//Points linked to the given DotMain point are described //by
the following class
```

```
class BoundDot
{public:
 CPoint DotP; // coordinates
 UINT DotT;   // type (label) of boundary coming out of a
            // DotMain point. This can change on the way.
 UINT Angle; // boundary angle (measured in degrees)
 UINT RegL, RegR; // auxilary variables
};



/* ==========
ArrayBound is an array storing pointers to boundary segments.
The junctions are joined by segments of region boundaries,
which are currently created by the user as part of the data
simulation procedure. A boundary can be of different geometric
types (a straight line, a broken  lines, or a  spline) and are
represented by a series of (interpolation) points that are
interpreted depending on the boundary type. It is assumed that
each boundary must start and end exactly at two (not
necessarily different) DotMain points.
===========*/
// This array stores points needed to calculate the
// boundary linking two junctions.
typedef CTypedPtrArray<CPtrArray, CPoint*> ArrayPoint;
class Bound
{public:
     UINT TBCreate; // geometric type of boundary
     UINT TypeBound; // auxilary variable
     ArrayPoint* ArrP;
};

typedef CTypedPtrArray<CPtrArray, Bound*> ArrayBound;
```

Appendix 2. Original and Presegmented Color Images