Declarative Approach of Inverse Lighting Problems

V. Jolivet, D. Plemenos, P. Poulingeas.
MSI Laboratory.
83 rue d’Isle, 87000 Limoges.
France.
E-mail: 
  jolivet@unilim.fr  
  plemenos@unilim.fr  
  poulinge@msi.unilim.fr

Abstract: In the last years, a lot a research have been done in the field of inverse lighting (and, in a neighbouring domain: inverse rendering – or relighting). However the proposed tools for inverse lighting are not intuitive or are too limited. Our aim is to present a framework for a declarative specification of the lighting of a scene. The concepts and properties relevant to a qualitative description of lighting will be presented. In a next phase, their expression in fuzzy subsets theory will be done. To finally getting solutions to our problem, a new inverse lighting method will be proposed, especially fitted for CSP solving techniques.

Keywords: Inverse Lighting, Declarative specification of lighting, Fuzzy logic, CSP.
1. Introduction.

With usual modellers, when a user wants to light a part of a scene to have a characteristic ambience, he must place himself the lights and give them their physical parameters (like luminous intensity) in a more or less intuitive manner. As the modeller doesn’t have any tools to help the user in this task, he wastes a lot of time before getting the wished lighting.

Inverse lighting algorithms try to find the main features of lights (position, orientation, physical parameters, etc.) from a set of lighting wishes given by the designer.

[PP01a], [PP01b] and [POU01] have presented the state of the art in the inverse lighting problem, using different criteria. One of the conclusions of these works is that when the user-interface of the software is intuitive enough, the possibilities of lighting specifications are very limited (For example, the light positions must belong to a set of positions fixed a priori), and, on the other hand, when the user can give some detailed lighting specifications, he must use a low-level software interface (He has to find some coefficients or to write scripts to express his wishes). Moreover, the inverse lighting algorithms frequently use optimization techniques that reject some configurations that the designer could estimate interesting (whereas he would have find them undesirable with only his first and vague idea of lighting).

The drawbacks of inverse lighting methods that [PP01a], [PP01b] and [POU01] point out are similar to those stressed by declarative modelling ([PLE95], [BON99]) in its analysis of traditional modelling. It seems then useful to build a new tool where the lighting design is described declaratively (with a linguistic description) and where many solutions are produced by the software.

Before talking about the main components of such a software, some useful situations for inverse lighting must be studied. Choosing one of these typical situations will have a big influence on our future tool because it will define its goals.

We will restrict ourselves to three majors situations:

1. The realistic lighting of a place (e.g. a room) as in architectural design. In this case, the lights must often follow some well-defined rules (They will lie at some particular places – like the ceiling, or they will be put in a row, etc.), and they generally can be seen (The user can explore the scene and lights can appear in his viewpoint).

2. The lighting of an event, like a museum exhibition of statues. Some elements have to be emphasized with a special lighting. The lights doesn’t have to follow the previous canons. The user can see the lights in the scene but it may be disturbing for him.

3. The lighting for a picture, to make an advertisement for example. The lights can have a lot of possible positions and physical characteristics because we are looking for an accurate ambience in a picture. Generally, the lights doesn’t appear in the final picture. Moreover, the user’s viewpoint is now very important. It is fundamental to emphasize an element.

In the two first situations, some lights are added in a scene which the user can explore after. In the third situation, the lighting parameters are set to produce an image. Hence, constraints and goals are very different, and the corresponding algorithms too.

Our recent works [JPP02] have focused on the two first problems. That’s
why radiosity is our illumination model. With radiosity, an user can explore a scene with a simple Z-buffer.

For our declarative tool, we propose the following scheme:

**Declarative Specification**

→ **Fuzzy Intervals**

→ **Constraint Satisfaction Problems**

→ **Inverse Lighting Problems**

→ **Rendering of the Scenes**

This is an idealistic scheme rather than a realistic one. In practice, the inverse lighting algorithm is applied just after the declarative specification, using some constraints on the lighting wished for some objects of the scene.

A comparison can be made between the different components of the previous scheme and the three main stages of declarative modelling ([PLE95], [BON99]).

The description phase of declarative modelling is similar to the declarative specification of the inverse lighting problem.

The generation phase is made here by the solving of the Constraint Satisfaction Problem and the algorithm solving the inverse lighting problem.

The scene understanding is limited to the rendering of the solution scenes. Learning mechanisms as these presented in [PRT98] haven’t been used yet in our works. These methods can reduce the number of produced scenes by cancelling many results containing undesirable characteristics for the designer.

We will now examine how the works made in [JPP02] can be included in our scheme for a declarative tool.

In [JPP02], a new inverse lighting algorithm has been introduced, based on a Monte-Carlo method. We have also explained how the constraints on the objects to be lit can be transformed in fuzzy intervals (following the ideas expressed in [DES95] and [DP97]). This last part of our paper will be continued here. Besides, an adaptation of the method exposed in [JPP02] will be developed to facilitate the solving of the Constraint Satisfaction Problem.

In section 2, the more interesting properties to help the designer to declaratively specify his lighting wishes will be studied. Then, in section 3, we will see how some properties studied in the previous section can be represented by fuzzy intervals, how modifiers can affect their membership function and how to obtain the intervals which will act as constraints for the next phase. A new inverse lighting algorithm will be detailed in section 4. It is based on the work of [JPP02], but it is more fitted to classical Constraint Satisfaction Problems algorithms. In section 5, we will study the resolution of a Constraint Satisfaction Problem made of the properties wished by the user and the solutions found by the new inverse lighting algorithm. An example will be developed in section 6, resuming the various stages of our scheme. Finally, we will conclude in section 7.


2.1. Introduction.

In this section, we are looking for lighting properties that can be described in a declarative manner. For that, the vocabulary from [DES95] will be used.

In this work, the properties corresponding to the objects to be lit will be distinguished from the ones corresponding to the lights (But there is
often a dependence between them in practice).

2.2. The lighting of the objects.

By “object”, we mean a set of patches of the scene, having a semantic unity stored in a file where a name is given to this set of patches.

Three basic properties can be found from this concept (i.e. lighting) : weakly, normally and strongly (e.g. “A is weakly lit”). The sentence “A is lit” can be understood as “A is normally lit”).

We can apply to these basic properties:

• a set of modifiers. Example : “A is enough little lit”.
• a set of fuzzy operators. Example : “A is more or less lit”.

Remark: A modifiable basic property is called by [DES95] a simple property. Under conditions, several modifiers and fuzzy operators can be applied to a simple property. The designer may want to specify “A is really very strongly lit”.

For parametric properties, they don’t seem relevant to this conceptual level. A description like “A has a lighting between 100 and 200 Watts” is managed by fuzzy intervals (A method that will be exposed in section 3).

However, simple comparison properties can be introduced. The designer may want to have some situations like “A is less lit than B”.

Finally, the negation of a simple property can be proposed. We can have a specification like “A is not very strongly lit”. In the next section, we will see that the interpretation of this type of sentence isn’t the simple logical negation of the property (cf. [DP97]).

Remark about the linguistic interpretation of the properties asked by the user:

Let’s consider a scene with two objects A and B, which are very near. We have the two following specifications:
- “A is very strongly lit” (1)
- “A is very strongly lit and B is very weakly lit” (2)

If the designer wants (1), doesn’t he expect implicitly (2) ? The problem is therefore the linguistic interpretation to give to the user’s wish. We could solve the ambiguity by giving to the user some propositions similar to (1), in the manner used by [DP97] to process the negation of a property. Obviously, it will be something interesting only if A and B are near enough (So, we need a criterion to decide if we have to use this method).

2.3. Properties on lights.

2.3.1. Introduction.

For the lighting, only one concept has been found. However, for the lights, a lot of concepts appear, geometrical and quantitative concepts. The shape of the light will have a big influence on the other concepts of this light. Moreover, the translation of properties in fuzzy intervals will be less intuitive than for the lighting of an object.

2.3.2. The concept of shape.

The shape of a light is an essential concept. If the designer can introduce some spotlights, some concepts relative to area lights won’t apply to spotlights, and vice-versa (For example the size, or orientation which has not the same meaning for a spotlight). The shape has a great influence on the properties of other concepts (like position, size and orientation). Then, a coarse definition of the shape implies many properties satisfied. Thus, a quadrilateral has less geometrical constraints than a square, and it can
produce more solutions for the simple property: “The light is extremely high”.

A finite domain $D$ can be associated with this concept of shape. $D$ will contain area lights and spotlights. At first, $D$ can be limited to the set: \{square, rectangle, parallelogram, quadrilateral, disk\} $\cup$ \{spotlight\}. There is a one-to-one correspondence between a property and a value of $D$. These properties can be processed as parametric properties with maybe a fuzzy approach for the shape of an area light.

2.3.3. Specific concepts and properties for area lights.

We have already talked about concepts in the previous subsection to stress the difference between area lights and spotlights.

Three properties can be related to the concept of size of an area source: small-sized, medium-sized and tall. We can apply to these properties some modifiers and fuzzy operators (the same as these viewed in subsection 2.2. for the lighting of an object). We can also introduce a comparison property between two lights (e.g.: “The light A is larger than the light B”) and a n-airy property (e.g.: “The light A is the largest”). Besides, introducing parametric properties is not very useful for this concept.

The orientation of the light can be an interesting concept for the designer. The orientation is given by reference to a direction parallel to the plane where is the light (typically a direction parallel to the sides of the polygon modelling the ceiling of a room). A parametric property can therefore be introduced (e.g.: “The angular value between the light A and the direction D is 45°”), and we can apply some fuzzy operators on it.

2.3.4. Specific concepts and properties for spotlights.

Two fundamental parameters can describe a spotlight (besides the position). These are the direction and the fall-off parameter.

These characteristics will correspond to two concepts. The associated properties will be parametric properties, with the ability for the designer to apply some fuzzy operators on them.

These parameters are strongly dependent on the lighting wished for the objects of the scene.

2.3.5. Concepts and properties common to all the types of lights.

The number of lights is an important concept, with a big influence on the low-level layers of a lighting software.

In relation with this concept, three basic properties can be found: little, middle and big. Parametric properties can be introduced too (e.g.: “The number of lights is 2”).

If some modifiers or fuzzy operators can be applied to basic properties, the wish of a parametric property must be processed accurately.

The alignment of the lights (or some of the lights) is a common property in architectural scenes. Alignment is processed as a basic property on the entire scene (or a part of the scene). When the designer expects the alignment of some lights, there is no ambiguity in his specification. However, a little degree of imprecision can be used to have some scenes when an accurate alignment isn’t possible.

The user can specify in the scene a zone for the lights. There is two types of properties corresponding to these wishes:

1. The software must satisfy the property (e.g.: “The lights are put at the ceiling of the room”).
2. The property is a vague and imprecise description of the light
position. However, the position of the light must also verify the first type of properties. The second type of properties have to be formulated in reference to the zones of the first one (if the designer uses this type of constraints), and specify the light position into these zones.

The overall of these two kinds of properties can lead to such a specification: “The light A is placed in the North of the ceiling”. This is equivalent to the conjunction of the following propositions:

1. “The lights are put at the ceiling”
2. “The light A is placed in the North”

The second type properties are basic properties, and modifiers or fuzzy operators can be applied on them (e.g.: “The light A is ‘very’ placed to the North” ⇔ “The light A is placed to the extreme North”).

These notions of regions can be represented on a map. The figure 1 is an example where the zone for the light position is rectangular:

3. Representing a property with fuzzy subsets.

3.1. Introduction.

[DES95] has proposed to represent a basic property not just as a standard interval, but as a fuzzy interval. This is to take into account the vagueness of words like “big” or “weak”. So, for the regions of the map of subsection 2.3.5., the frontiers cannot be precisely given.

With this hypothesis, [DES95] has studied the application of modifiers and fuzzy operators to the membership function associated to a basic property.

In [DP97] is developed a linguistic interpretation of the negation of a property. This negation cannot be formed as the logical negation of the property. The designer rather thinks to a similar property in the same domain (The notion of similarity between two fuzzy subsets is detailed in [DP97]). A set of choices is then proposed to the user, with the best ones at first.

These works take place in declarative modelling. Their obvious application is the verification of the generated scenes. Hence, the fuzzy intervals representation will happen at the end of the generation phase.

On the contrary, in our resolution engine, we want to use fuzzy intervals at the beginning of the generation phase. Some changes are then necessary.

[DES98] has proposed too a generation method based on fuzzy intervals (the recursive generation) in his CordiFormes project. It’s an universal technique, but coming with a scene modelling ontology and not especially fitted for inverse lighting problems.

3.2. Getting a constraint from a fuzzy interval.

To define the membership function f associated to a property, [DES95] uses a
quadruple \((\alpha, a, b, \beta)\) and two functions \(L\) et \(R\) called form functions. \(f\) is then a membership function of \(L\)-\(R\) type. [DES95] chooses some form functions with interesting properties for \(f\), especially:

- \([a,b]\) is the kernel of the membership function \(f\).
- \([a-\alpha, b+\beta]\) is the support set of the membership function \(f\).
- \((\alpha>0\ or\ \beta>0\ and\ a\neq b) \iff\) The fuzzy subset given by \(f\) is a fuzzy interval.

The membership function \(f\) is:

\[
f : D \rightarrow [0,1] \\
t \rightarrow 0 \text{ if } t < a - \alpha \\
L((a-t)/\alpha) \text{ if } \alpha \neq 0 \text{ and } (a-\alpha \leq t < a) \\
1 \text{ if } a \leq t \leq b \\
R((t-b)/\beta) \text{ if } \beta \neq 0 \text{ and } (b < t \leq b+\beta) \\
0 \text{ if } t > b + \beta
\]

When an \(\alpha\)-support set \(A_v\) is computed for a real value \(v \in ]0,1]\), we have:

\[
A_v = [a - \alpha L^{-1}(v) ; b + \beta R^{-1}(v)]
\]

We try to find the boundaries of this interval to build a constraint. So these boundaries should be easily computed (with some simple formulas for \(L^{-1}\) and \(R^{-1}\)). [DES95] doesn’t have this problem, because checking the fit of a scene needs only to check if a value \(t\) of the domain \(D\) has a membership degree superior to \(v\).

[DES95] gives some examples of interesting functions \(L\) and \(R\). The most used functions in fuzzy subsets theory are trapezoidal functions defined by:

\[
\forall x \in \mathbb{R}, L_t(x) = R_t(x) = \max(0, 1-x)
\]

As reciprocal functions are limited to the interval \([0,1]\), obviously we have:

\[
L_t^{-1}(x) = R_t^{-1}(x) = 1 - x
\]

Remark: Functions with easily computable reciprocals are the most interesting. If it’s not possible to formally determine the reciprocal function, the domain can be discretized. The values given by this discretization can then be tested in an incremental manner, but it is a more costly method than direct computation of the \(\alpha\)-support set.

For parameters \(a, b, \alpha\) and \(\beta\), they might be given by psycho-physical tests. For the other properties (like the size or the position of a light), the values of \(a, b, \alpha\) and \(\beta\) will depend on the geometry of the scene.

Remark: Some properties can be easily represented by some membership functions a little bit different from \(L\)-\(R\) functions. Let’s study the property “North” (i.e. placed to the North) of a position light. Let’s call \(l\) the width and \(h\) the height of the map of the subsection 2.3.5. If South-West is the origin and \((x,y)\) the cartesian coordinates of a point, the membership function will be:

\[
f : D= [0,l] x [0,h] \rightarrow [0,1] \\
(x,y) \rightarrow 1 \text{ if } y \in [2h/3 ; h] \\
(3/h)*y-1 \text{ if } y \in [h/3 ; 2h/3] \\
0 \text{ if } y \in [0 ; h/3]
\]

The \(\alpha\)-support set will be (for \(v \in ]0,1]\)):

\[
A_v = [0 ; l] x [(h/3)*(v+1) ; h]
\]

3.3. Action of a modifier on the membership function representing a property.

3.3.1. Introduction.

[DES95] and [DP97] have developed some techniques to determine the membership function \(f'\) associated to a property like “\(x\ is\ m\ P\)” where \(m\) is a modifier and \(P\) a simple property. The function \(f'\) depends on the membership function \(f\) associated to \(P\) and the modifier \(m\), but also on other semantic parameters of the property \(P\).

In practice, the application of a modifier makes a translation and a contraction (or a dilatation) on the
membership function. For our work, the most important thing is the facility (or difficulty) to compute quickly an \( \alpha \)-support set with the new membership function \( f' \).

3.3.2. The new membership function.

The basic translation coefficient of the property \( P \) will be noticed \( t \) (\( t \) depends on the semantic and the sign of \( P \)). The modification coefficient of the modifier \( m \) will be noticed \( k \) (\( k \) depends on the sign of \( m \) and its distance to the modifier \( \emptyset \)). The membership function \( f' \) associated to the property \( mP \) is characterised by the formulas:

\[
\begin{align*}
\alpha' & = \alpha^* (1-|k|*10\%) \\
\beta' & = \beta^* (1-|k|*10\%) \\
a' & = a + k^* t + \frac{1}{2} |k^* (b-a)*10\%| \\
b' & = b + k^* t - \frac{1}{2} |k^* (b-a)*10\%| \\
t' & = \text{sign}(k^*) \text{sign}(t^*) |t^*| \\
k' & = k \\
L' & = L \\
R' & = R
\end{align*}
\]

The form function does not change, so the formula for the \( \alpha \)-support set given in subsection 3.2. holds.

3.4. Action of a fuzzy operator on the membership function representing a property.

Let’s consider now a property like “\( X \) is \( o \) \( P \)” where \( o \) is a fuzzy operator. A contraction (or a dilatation) will be applied to the membership function \( f \) of \( P \). However, the kernel won’t change.

A fuzzy operator is characterised by a fuzzy coefficient \( j \in \mathbb{R}^* \) which do not depend on the semantic of the property \( P \).

The membership function \( f' \) representing the property \( oP \) has as fundamental parameters:

\[
\begin{align*}
\alpha' & = \alpha^* (1-j*10\%) \text{ if } j \in ]1,+\infty[ \\
\beta' & = \beta^* (1-j*10\%) \text{ if } j \in ]1,+\infty[ \\
\beta^* (1+(1/j)*10\%) \text{ if } j \in ]0,1[ \\
a' & = a \\
b' & = b \\
t' & = t \\
k' & = k \\
L' & = L \\
R' & = R \\
\end{align*}
\]

For an \( \alpha \)-support set \( A_\alpha \) associated to the function \( f' \), the formula will become:

\[
A_\alpha = [a-\alpha' L'^{-1}(v) ; b+\beta' R'^{-1}(v)]
\]

where:

\[
L'^{-1}(v)=R'^{-1}(v)=(1-v)^{1/j}
\]


A Monte Carlo-based inverse lighting method was proposed in [JPP02].

A little number of rays are shot from the patches to be lit. The hit patches are stored in a list, called \( L \). These patches can be lights. Rays are shot during several stages, until the list \( L \) doesn’t increase much. We suppose then that we have got in \( L \) the most of the patches able to lit directly the goals expressed by the designer. Generally, this list is very big. Following the optimisation techniques used in inverse lighting problems, a form factor-based heuristic has been introduced to let in \( L \) the most important patches. Finally, the convex hull of the vertices of the patches belonging to \( L \) was computed. We suggest to place one or more lights into this computed zone after its discretization. Here is an example of a scene got with this algorithm (figure 2):
However, a drawback of this method is the intricacy induced for the Constraint Satisfaction Problem. Indeed, the constraint on the emitted radiance must be satisfied at last (after these on position, size, etc.). It’s then a very important constraint, and the need to process it after all the others prevent to use classical optimisations from CSP solving techniques. Hence, we propose a new algorithm, better fitted for constraint satisfaction problems.

First, the parts of the scene formed by the elements which could be lights will be uniformly remeshed. A new mesh is not useful for all the elements because there may be some constraints on the light position (The lights may have to be put at the ceiling by example). The new mesh will be done with squares, of an enough little size. As in [JPP02], there are several ray-casting stages. However, we store now in a list $L_i$ ($i$ is a number of a patch to be lit) not only the numbers of the patches hit by the rays shot from the patch $i$ but also an interval for the valid emitted radiance. After processing all the patches to be lighted, we have a list $L$ with couples of patches and intervals.

If the wishes of the designers are contradictory, we will have:

$$L = \emptyset$$

By example, it will be produced when the user will want to light two near objects with very different lighting specifications.

The light will then be a combination of the patches belonging to the list $L$.

5. Solving a constraint satisfaction problem.

With the new algorithm of research of the potential emissive patches presented in the previous section, the constraints on the lighting wished by the user for the objects of the scene are necessarily satisfied (because the algorithm is provided for). The constraint satisfaction problem that must be solved is only formed with constraints on lights. However, the sweep of the intervals determining the emitted radiance must be done after their discretization, to generate several solution scenes.

For an area light, the problem has been reduced to the selection of the size, the orientation and the position. The constraint satisfaction problem can be processed with algorithms like Lookahead [TSA94] or learning by failure (which allows an intelligent backtracking, like Conflict Directed Backjumping [PRO93] or Nogood Recording [SCV94]). When the shape and the orientation have been fixed, the only thing to do is to place the light in the zone corresponding to the list $L$ computed in section 4. That’s why we can have some gains with the CSP solving methods.

If the number of lights is not fixed by the user, the process of this number can be done in an incremental manner. The drawback of this method is to give to the user a lot of solution scenes with the same number of lights before this number increases. One possibility to overcome this problem is to use PLC [IMB96] where the number of variables can dynamically evolve.
6. An example.

The previous ideas will be now illustrated with the following scene (figure 3):

![Fig. 3. – An indoor scene.](image)

A designer gives the following specification:
- The top of the table is very strongly lit.
- The lights are placed at the ceiling.
- The shape of the lights is a square.
- The number of lights is 1.
- The size of the light A is enough little small.

The lighting of the table will be studied first.

As fundamental parameters for the basic property “strongly lit”, we can take:
\[ a = 100 \text{ Watts} \]
\[ b = 150 \text{ Watts} \]
\[ \alpha = 10 \]

Hence, the membership function \( f \) associated to the previous property is:
\[
f(t) = \begin{cases} 
0 & \text{if } t<90 \\
t/10-9 & \text{if } 90\leq t<100 \\
1 & \text{if } 100\leq t\leq 150 \\
16-t/10 & \text{if } 150\leq t\leq 160 \\
0 & \text{if } t>160 
\end{cases}
\]

The property wished by the user is: “very strongly lit”. The modification coefficient of \( m \) can be \( k=2 \), and the translation coefficient \( t=20 \).

The fundamental parameters for the simple property “very strongly lit” will then be:
\[ a' = 145 \text{ Watts} \]
\[ b' = 185 \text{ Watts} \]
\[ \alpha' = \beta' = 8 \]

The associated membership function \( f' \) will be:
\[
f'(t) = \begin{cases} 
0 & \text{if } t<137 \\
1-(145-t)/8 & \text{if } 137\leq t<145 \\
1 & \text{if } 145\leq t\leq 185 \\
1-(t-185)/8 & \text{if } 185\leq t\leq 193 \\
0 & \text{if } t>193 
\end{cases}
\]

The representative curves of \( f \) and \( f' \) are (cf. figure 4):

![Fig. 4. – Representative curves of the membership functions \( f \) and \( f' \).](image)

If we take as acceptance threshold the value 0.5, we will have the following \( \alpha \)-support set:
\[ A_{0.5} = [139,189] \]

For each patch \( i \) of the table, the power \( \Phi_i \) must have a value such that:
\[ 139 \leq \Phi_i \leq 189 \]

Now, let’s examine the size of the light.

As fundamental parameters for the property “little small”, we propose:
\[ a = 0.5 \]
\[ b = 1.5 \]
\[ \alpha = \beta = 0.25 \]

The membership function \( f \) associated to this property is:
The property desired by the designer is: “enough little small”. The modification coefficient of \( m \) can be in this context \( k=1 \), and the translation coefficient \( t=0.1 \).

The parameters for the property “enough little small” will then be:
\[
\begin{align*}
a' &= 0.65 \\
b' &= 1.55 \\
\alpha' = \beta' &= 0.225
\end{align*}
\]

The associated membership function \( f' \) will be:
\[
\begin{align*}
f'(x) &= 0 \quad \text{if } x < 0.425 \\
x/0.225-1 \quad \text{if } 0.425 \leq x < 0.65 \\
1 \quad \text{if } 0.65 \leq x \leq 1.55 \\
7.9-x/0.225 \quad \text{if } 1.55 < x \leq 1.775 \\
0 \quad \text{if } x > 1.775
\end{align*}
\]

The representative curves of \( f \) and \( f' \) is then (cf. figure 5):

![Membership functions](image)

With an acceptance threshold of value 0.5, we will have the following \( \alpha \)-support set:
\[
A_{0.5} = [0.5375, 1.6625]
\]

Let’s 1 be the side of a patch belonging to the ceiling. If we have \( l=0.1 \), a side of the light will be formed with a number of patches belonging to the interval:
\[
[5.375, 16.625] \cap \mathbb{N} = [6, 16] \cap \mathbb{N}
\]

7. Conclusion.

In this paper, we have seen the different components of a tool for a declarative description of lighting. The major concepts for a specification of lighting have been studied and translated into fuzzy subsets theory. A new algorithm for inverse lighting, adapted from [JPP02], has been presented to make easier the CSP solving phase. In the future, we will code these ideas in our prototype (already used in [JPP02]) and will study the human perception in the aim of obtaining a mathematical framework for a qualitative description of the ambience of a scene.

References:


