Computer Science Journal of Moldova, vol.**16**, no.1(46), 2008

Some Remarks on Blowing-Ups in a Computer Algebra System

Anne Frühbis–Krüger

Abstract

The aim of this short note is to provide detailed information on how to compute blowing ups in various settings by means of a computer algebra system. All examples are formulated using the system SINGULAR[5].

1 Introduction

Although the notion of blowing up is ubiquitous in algebraic geometry and singularity theory, the most common use of it is a blowing-up at a point. Consequently tools to compute blowing-ups at a point are implemented in a wide range of computer algebra systems.

For more complex applications on the other hand like e.g. studying examples of flops, considering Nash modifications or desingularization, restricting the choice of centers to sets of points is not an option: Considering the even the simplest example of a flop, the centers of the blow-ups for the small resolutions will be Weil divisors[1]; considering resolution of singularities, the singular locus has, in general, no reason to be zerodimensional. Thus implementations of blowing ups also need to cover the general case, which will be described in section 2 of this article. Allowing centers to be higher dimensional, we encounter problems of efficiency, which may be countered to some extent by considering embedded blowing up, whenever the centers are non-singular and we are using a covering by affine charts; allowing centers to even

[1]see section 3 for the step-by-step calculations in SINGULAR

be singular, as is necessary for Nash modifications, on the other hand, blocks this alternative for computations.

In sections 3 and 4, we consider examples of applications of the various variants mentioned – each time including a step by step SINGULAR-input and output for treating the respective task.

As usual for computational methods for algebraic geometry in characteristic zero, we assume the ground field to be $\mathbb{C}$ for all reasoning, although actual computations are performed over the rationals.

I should like to thank Lawrence Ein, Priska Jahnke, Patrick Popescu-Pampu, David Ploog and Ivo Radloff, whose questions on how to compute specific examples of blowing ups by means of a computer algebra system, led to this collection of remarks. I am also indebted to the Freie Universität Berlin and the ICTP Trieste for the invitations that provided the opportunity to meet the previously mentioned colleagues.

2 Implementation of Blowing Ups

2.1 Blowing Up a Scheme

Recall that given a noetherian scheme W and a closed subscheme Y of W, corresponding to a ideal sheaf $\mathcal{I}$ on W, the blowing-up of W along Y is defined as

$$\pi : \tilde{W} := Proj(\bigoplus_{d \geq 0} \mathcal{I}^d) \longrightarrow W.$$

This is a birational map, which is an isomorphism away from Y, i.e. $\tilde{W} \setminus \pi^{-1}(Y) \cong W \setminus Y$; the inverse image $\pi^{-1}(Y)$ is a Cartier divisor on $\tilde{W}$, called the exceptional divisor of the blowing up[2]. Unfortunately, this description is not well-suited for explicit calculations and implementations, which usually require objects to be represented by polynomial data, i.e. a free presentation or a set of generators of an ideal over a polynomial ring or a quotient thereof. To achieve this description, a

[2]For further details including the universal property of blowing up see any textbook on algebraic geometry like [6], II or [9]

convenient way is to pass to a covering of W by finitely many affine open sets. Then the initial situation in an affine chart $U \subset W$ can be formulated as follows: Working over the basering $A := \mathcal{O}_W(U)$, which is a polynomial ring or a quotient thereof, we can describe the center Y in this chart by the ideal $I = \mathcal{I}(U)$ which we now assume to be generated by $f_0, \ldots, f_s$. Then we can consider the graded morphism of A-algebras

$$\begin{aligned} \varphi : A[y_0, \ldots, y_s] &\longrightarrow A[t \cdot f_0, \ldots, t \cdot f_s] \subset A[t] \\ y_i &\longmapsto t \cdot f_i \end{aligned}$$

whose image is obviously isomorphic to $\bigoplus_{d \geq 0} I^d$. Hence $\pi^{-1}(U)$, as a subset of $U \times \mathbb{P}^s$, allows a description by $A[y_0, \ldots, y_s]/ker(\varphi)$ which is precisely what we needed for computational purposes. The exceptional divisor of the blowing up, i.e. the inverse image of the center Y, then corresponds to the ideal $I \cdot A[y_0, \ldots, y_s]/ker(\varphi)$.

The ideal $ker(\varphi)$ unfortunately involves $s + 1$ additional variables and hence it seems at first glance that e.g. the number of variables in the resolution process might constantly rise making effective standard bases calculations virtually impossible after just a few blow-ups. But passing once again to an appropriate affine covering helps us keep the number of variables sufficiently low; more precisely we use the usual covering of the newly introduced $\mathbb{P}^s$ by the sets $D(y_i)$, $1 \leq i \leq s$. Obviously, this is a trade-off and causes the calculations to branch which easily leads to duplicate calculations on the intersections of several charts and significantly increases the amount of data to be stored. On the other hand, treating several charts at the same time on different processors/computers allows a parallelization of e.g. the resolution algorithm which is only rarely possible for computational tasks in commutative algebra and may improve the performance.

Nevertheless, the disadvantages of passing to open covers largely outweigh the benefits in general and it is therefore desirable to keep the number of charts as low as possible e.g. by dropping charts which do not contribute any new information to the considered task.

Example 1 [Blowing up of $\mathbb{A}^3$ at the origin]

```
ring r=0,(t,x(1..3),y(1..3)),(dp(1),dp);
                      // A^3 x P^2 plus extra variable
                      // for elimination of t
                      // as usual in computation of
                      // preimage of zero
ideal I=y(1)-t*x(1),
        y(2)-t*x(2),
        y(3)-t*x(3);  // ideal describing map
ideal IW=eliminate(I,t);
                      // elimination step
ring r2=0,(x(1..3),y(1..3)),dp;
                      // A^3 x P^2
ideal IW=imap(r,IW);  // transfer the ideal to this ring
IW;                   // ideal of variety after blowing up
--> IW[1]=x(3)*y(2)-x(2)*y(3)
--> IW[2]=x(3)*y(1)-x(1)*y(3)
--> IW[3]=x(2)*y(1)-x(1)*y(2)

subst(IW,y(1),1);     // what does the chart
                      // D(y(1)) look like
--> _[1]=x(3)*y(2)-x(2)*y(3)
--> _[2]=-x(1)*y(3)+x(3)
--> _[3]=-x(1)*y(2)+x(2)
// As expected this is isomorphic to an A^3, getting rid
// of x(2) and x(3) using generators _[2] and _[3].
// The exceptional divisor is described by x(1)=0 in
// this chart.
//
// The same observations hold in the other charts,
// as the whole situation is blind to exchanging the roles
// of the variables x(i).
```

As already mentioned, we would like to blow up at more general centers than point. Here is one such example:

Example 2 [Blowing up $\mathbb{A}^3$ in $V(z, x^2 + y^2 - 1)$]

```
ring r=0,(t,x(1..3),y(1..2)),(dp(1),dp);
                      // A^3 x P^1 plus extra variable
                      // for elimination of t
                      // as usual in computation of
                      // preimage of zero
ideal I=y(1)-t*x(3),
        y(2)-t*(x(1)^2+x(2)^2-1);
                      // ideal describing map
ideal IW=eliminate(I,t);
                      // elimination step
ring r2=0,(x(1..3),y(1..2)),dp;
                      // A^3 x P^1
ideal IW=imap(r,IW);  // transfer the ideal to this ring
IW;                   // ideal of variety after blowing up
--> IW[1]=x(1)^2*y(1)+x(2)^2*y(1)-x(3)*y(2)-y(1)

subst(IW,y(1),1);     // what does the chart
                      // D(y(1)) look like
--> _[1]=x(1)^2+x(2)^2-x(3)*y(2)-1
// This is obviously non-singular, but we cannot get rid
// of a fourth variable.
subst(IW,y(2),1);     // and D(y(2)) -->
_[1]=x(1)^2*y(1)+x(2)^2*y(1)-x(3)-y(1)
// Here we can get rid of x(3).
```

This sequence of computational steps to compute blowing ups is available as commands `blowUp` and `blowUp2` in SINGULAR, see the SINGULAR online manual for a description.

2.2 Notions of Transforms

Considering blowing ups, we are usually not just dealing with a single scheme, but additionally with one or several subschemes of it which are also affected by the blowing up. This leads to the task of computing the total and the strict transform of such a subscheme (or depending on the context also the weak or the controlled transform).

To this end, let us recall that the total transform of a closed subscheme $Z \subset W$ (corresponding to an ideal sheaf $\mathcal{J}_Z \subset \mathcal{O}_W$) under the blowing up π is just the inverse image $\pi^{-1}(Z)$ and can hence be computed as

$$\mathcal{J}_{Z,total} = \mathcal{J}_Z \cdot \mathcal{O}_{\tilde{W}}.$$

Let us further recall that the strict transform $\tilde{Z}$ of Z is obtained by blowing up Z at the center given by $\mathcal{I} \cdot \mathcal{O}_Z$ according to the following commutative diagram:

$$\begin{array}{ccc} \tilde{Z} & \stackrel{i}{\hookrightarrow} & \tilde{W} \\ \downarrow & & \downarrow \pi \\ Z & \stackrel{i}{\hookrightarrow} & W. \end{array}$$

In the affine case, we can also obtain the strict transform of Z by forming the closure of $\pi^{-1}(Z \setminus (Z \cap Y))$ in $\tilde{W}$. By using again the previously introduced affine covering of $\tilde{W}$, this allows us to compute the strict transform from the total transform using a saturation[3]:

$$J_{Z,strict} = (J_Z \cdot \mathcal{O}_{\tilde{W}}(U) : I_E^{\infty}),$$

where J_Z is used as short hand notation of $\mathcal{J}_Z(U)$ and I_E denotes the ideal of the exceptional divisor of π on our chart U. Geometrically this saturation can be interpreted as dropping all components of the total transform which lie in the exceptional divisor or coincide with it.

For resolving singularities by the algorithmic approaches of Villamayor [1] and Encinas/Hauser [3] two other notions of transforms come into play which amount to ending the above saturation prematurely after a fixed number of ideal quotient computations. In the case

[3]Saturating, i.e. iterating the ideal quotient until it stabilizes (noetherian ring), is available in most computer algebra systems for algebraic geometry and commutative algebra as a built-in command. It is usually an expensive operation, but not if we are saturating by a principal ideal. For a detailed discussion of saturation and its geometric interpretation see [2] or [4]

of the weak transform, this number of iterations is the maximal order of the ideal J_Z prior to the blowing up (at a center contained in the locus of maximal order); geometrically speaking, the weak transform originates from the total transform by removing all copies of the exceptional divisor, but keeping the lower-dimensional components which lie inside the exceptional divisor. In the case of the controlled transform, the number of iterations is prescribed by the resolution algorithm and can be anything between 1 and the number of iterations for the weak transform.

Example 3 [Different notions of transforms of a space curve] Continuing the first example, we now consider the space curve $V(xz, yz, x^3 - y^4) \subset \mathbb{A}^3$ and compute its different transforms:

```
ideal J=x(1)*x(3),x(2)*x(3),x(1)^4-x(2)^3;
                               // ideal of space curve
ideal Jtotal=J,IW;             // ideal of total transform,
                               // before passing to charts
ideal Jt1=subst(Jtotal,y(1),1);
                               // ideal in chart D(y(1))
Jt1;
--> Jt1[1]=x(1)*x(3)
--> Jt1[2]=x(2)*x(3)
--> Jt1[3]=-x(1)^4+x(2)^3
--> Jt1[4]=x(3)*y(2)-x(2)*y(3)
--> Jt1[5]=-x(1)*y(3)+x(3)
--> Jt1[6]=-x(1)*y(2)+x(2)
// Obviously we can get rid of x(2) and x(3) by appropriate
// reductions. As the heuristic to do this automatically is
// lengthy, it is not printed here. Instead, we use our
// knowledge of what we want to replace:
ideal Jt2=subst(Jt1,x(3),x(1)*y(3));
                               // replace x(3) by x(1)*y(3)
                               // according to Jt1[5]
Jt2=subst(Jt2,x(2),x(1)*y(2));
                               // replace x(2) by x(1)*y(2)
```

```
                              // according to Jt1[6]
Jt2=interred(Jt2);            // drop unnecessary
                              // generators
Jt2;
--> Jt2[1]=x(1)^2*y(3)
--> Jt2[2]=x(1)^3*y(2)^3-x(1)^4

ring chart=0,(x(1),y(2),y(3)),dp;
ideal Jt2=imap(r2,Jt2);       // only keep necessary
                              // variables for this chart,
                              // by passing to appropriate
                              // ring
Jt2;
--> Jt2[1]=x(1)^2*y(3)
--> Jt2[2]=x(1)^3*y(2)^3-x(1)^4

ideal Jctrl1=quotient(Jt2,ideal(x(1)));
                              // controlled transform,
                              // #iterations=1
Jctrl1;
--> Jctrl[1]=x(1)*y(3)
--> Jctrl[2]=x(1)^2*y(2)^3-x(1)^3

ideal Jweak=quotient(quotient(Jt2,ideal(x(1))),
                     ideal(x(1)));
                              // weak transform
                              // #iterations=2, because
                              // ord(J)=ord(x(1)*x(3))=2
Jweak;
--> Jweak[1]=y(3)
--> Jweak[2]=x(1)*y(2)^3-x(1)^2

LIB"elim.lib";                // saturation is in elim.lib
ideal Jstr=sat(Jt2,x(1));
                              // strict transform
```

```
Jstr;
[1]:
   _[1]=y(3)                        // ideal of strict transform
   _[2]=y(2)^3-x(1)
[2]:
   3                                // number of iterations when
                                    // stabilizing
```

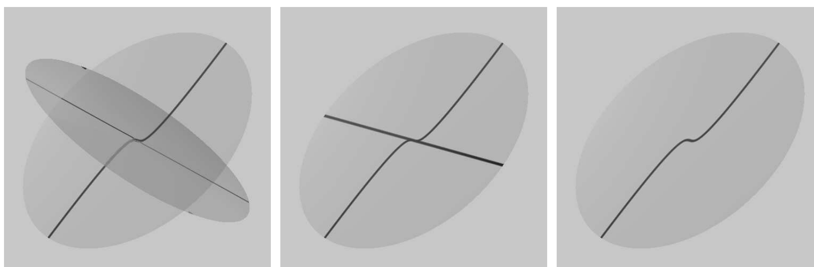

Figure 1. These three pictures illustrate the different notions of transforms computed in the example 3. From left to right, we see total transform, weak transform and strict transform. Due to technical reasons with the imaging tool surf, one additional plane is shown in each image: V(y(3)), of which we know that it contains the two curves.

The above considerations about the definition and the computation of the strict transform of a subscheme also imply that there are two equivalent ways of computing the blowing up of a scheme which can be embedded into a $\mathbb{A}^k$ or a $\mathbb{P}^k$ at a non-singular center:

- blowing up the scheme directly
- considering the scheme as embedded in an appropriate $\mathbb{A}^k$ (possibly after passing to an affine covering), blowing up the $\mathbb{A}^k$ and computing the strict transform

The first variant can be quite expensive in the elimination of the additional variables – depending on how complicated the equations for the variety are[4]. The latter variant has to deal with a larger amount of data due to the affine covering; the expensive part here is the saturation which is, on the other hand, cheaper than a general saturation, because we saturate by a principal ideal.

If the center itself is singular, however, blowing up the ambient space is not an option, because the ambient space has no reason to be smooth after such a blowing up as the following example shows:

Example 4 [Blowing up at a singular center]

```
ring r=0,(t,x(1..3),y(1..2)),(dp(1),dp);
                                  // again A^3 x P^1 plus
                                  // additional variable t
ideal I=y(1)-t*x(1)*x(2),y(2)-t*x(3);
                                  // center is the union
                                  // of the x- and y-axes
ideal IW=eliminate(I,t);
IW;
--> IW[1]=x(1)*x(2)*y(2)-x(3)*y(1)
subst(IW,y(2),1);                 // chart D(y(2))
--> _[1]=x(1)*x(2)-x(3)*y(1)
// this obviously has a singular point at the origin!
```

3 Application 1: A Flop

As the first application, we now consider the simplest example of a flop. It is however beyond the scope of this short note to explain exactly what a flop is; a good reference for the minimal model program (the context, in which the notions of flips and flops arose) and the precise definitions of flips and flops can be found in [7]. For our purpose here, which is

[4] Standard basis calculations w.r.t. elimination orderings are never really cheap, but (like standard basis calculations in general) they tend to become very expensive, if we are dealing with many variables and the equations are not of a particularly simple form.

an illustration on how to use a computer algebra system to deal with a concrete example, it suffices to consider the following situation:

Let $X = V(x_1x_2 - x_3x_4) \subset \mathbb{A}^4_{\mathbb{C}}$, which obviously has one isolated singularity at the origin. Blowing up X at this singular point, we obtain $\tilde{X} \subset \mathbb{A}^4_{\mathbb{C}} \times \mathbb{P}^3_{\mathbb{C}}$ whose exceptional locus turns out to be a $\mathbb{P}^1 \times \mathbb{P}^1$. On the other hand, blowing up at the Weil divisor $V(x_1, x_3) \subset X$ which is not $\mathbb{Q}$-Cartier, we obtain $X_1 \subset \mathbb{A}^4_{\mathbb{C}} \times \mathbb{P}^1_{\mathbb{C}}$. Analogously, blowing up at $V(x_2, x_4)$ yields another scheme X_2. Here it is interesting to observe that X_1 and X_2 may alternatively be constructed by blowing down one (and the other resp.) of the $\mathbb{P}^1_{\mathbb{C}}$ of the exceptional divisor of $\tilde{X}$. The resulting rational map $X_1 \dashrightarrow X_2$ is the well-known simplest example of a flop[5]. As a diagram, we have the following situation

$$\begin{array}{ccc} & \tilde{X} & \\ \swarrow & & \searrow \\ X_1 & \dashrightarrow & X_2 \\ \searrow & & \swarrow \\ & X & \end{array}$$

The following sequence of SINGULAR commands mimics the main steps of the above construction:

```
ring r=0,(t,x(1..4)),dp;    // A^4 plus extra variable t,
                            // for checking singular locus,
                            // Weil divisors, not Cartier;
                            // extra variable t will be
                            // needed later on -
                            // explained there
ideal I=x(1)*x(2)-x(3)*x(4);
LIB"sing.lib";              // slocus is in sing.lib
std(slocus(I));             // ideal of singular locus
```

[5]Considered abstractly, the two varieties X_1 and X_2 are isomorphic in this very simple example. This is a coincidence and does not occur in general.

```
--> _[1]=x(4)
--> _[2]=x(3)
--> _[3]=x(2)
--> _[4]=x(1)              // as we expected

ideal IDiv1=x(1),x(3);     // first divisor
ideal IDiv2=x(2),x(4);     // second divisor
// as both V(IDiv1) and V(IDiv2) are obviously reduced,
// irreducible, closed subsets of A^4, it remains to check
// - V(IDiv1) contained in V(I) and of codimension 1
// - analogously for V(IDiv2) -- not shown here
size(reduce(I,std(IDiv1)));
                           // zero if ideal containment
                           // test succeeds
--> 0
dim(std(I))-dim(std(IDiv1));
                           // codimension of V(IDiv1)
                           // in V(I)
                           // remark: extra variable t
                           // causes both dimensions
                           //        to be raised by 1
                           //        which does not
                           //        affect this result
--> 1
// Hence we have prime divisors on X, which are of course
// Weil divisors.
//
// We now check that V(IDiv1) cannot be Q-Cartier, i.e.
// that there cannot be a power of V(IDiv1) which is
// locally principal. To this end, we pass to the
// localization at the only singular point. - If it
// fails there, this is sufficient to show that V(IDiv1)
// is not Q-Cartier.
ring rlocal=0,(t,x(1..4)),(dp(1),ds);
                           // ds ordering is local!
```

```
def I=imap(r,I);
ideal Itest=I,x(1),x(3)*t-1;
reduce(1,std(Itest));      // 0, if some power of x(3) is
                           // in I+<x(1)>; 1 otherwise
--> 1
Itest=I,x(3),x(1)*t-1;     // as above, but roles of x(1)
                           // and x(3) exchanged
reduce(1,std(Itest));
--> 1
// This implies that V(IDiv1) cannot be Q-Cartier.
//
// After checking the claimed properties of $X$, we now
// return to blowing up and blowing down.
ring r2=0,(t,x(1..4),u(1..4)),(dp(1),dp);
                           // for A^4 x P^3 + extra
                           // variable
ideal Ipt=x(1)*x(2)-x(3)*x(4),u(1)-t*x(1),u(2)-t*x(2),
                              u(3)-t*x(3),u(4)-t*x(4);
                           // ideal for blowing up point
ideal IWeil1=x(1)*x(2)-x(3)*x(4),u(1)-t*x(1),u(3)-t*x(3);
                           // for Weil-divisor V(x_1,x_3)
ideal IXtop=eliminate(Ipt,t);
                           // I blownup at point
size(IXtop);
--> 9                      // 9 generators

std(IXtop+ideal(x(1..4))); // ideal of except.locus
--> _[1]=x(4)
--> _[2]=x(3)
--> _[3]=x(2)
--> _[4]=x(1)
--> _[5]=u(1)*u(2)-u(3)*u(4)   // <-- P^1 x P^1 in P^3

ideal IX1=eliminate(IWeil1,t);
                           // I blown up at first
```

```
                              // Weil divisor
IX1;
--> IX1[1]=x(3)*u(1)-x(1)*u(3)
--> IX1[2]=x(2)*u(1)-x(4)*u(3)
--> IX1[3]=x(1)*x(2)-x(3)*x(4)

// Now we blow down contracting the P^1 specified by
// V(u(2),u(4)) to a point
// In general this can only be done, if the corresponding
// blow-up map is known - it is then a preimage
// calculation.
// Here, however, the situation is so simple that we can
// see that this contraction amounts to a projection to
// A^4 x P^1, i.e. to eliminating u(2) and u(4)
eliminate(IXtop,u(2)*u(4));
--> _[1]=x(3)*u(1)-x(1)*u(3)
--> _[2]=x(2)*u(1)-x(4)*u(3)
--> _[3]=x(1)*x(2)-x(3)*x(4)
// As expected this is the same as IX1.
```

4 Application 2: A Nash Modification

As the second application, we consider Nash modifications, which are known to locally be blowing ups. In particular, we shall consider two examples, only one of which is a complete intersection.

Let us recall that given a reduced separated algebraic scheme X of pure dimension r, a Nash modification $p : \tilde{X} \longrightarrow X$ is defined by the following process (which, for simplicity of presentation, we describe only in the special case that $X \subset \mathbb{A}^n$ and that X is defined by $\langle f_1, \ldots, f_m \rangle$):

Denoting by G_r^n the Grassmannian of r-planes in $\mathbb{A}^n$, by $Reg(X)$ the complement of the singular locus of X, and by $T_{X,x}$ the tangent space of X at a point $x \in Reg(X)$, consider the morphism

$$\eta : Reg(X) \longrightarrow X \times G_r^n \qquad (1)$$

$$x \longmapsto (x, T_{X,x}). \qquad (2)$$

$\tilde{X}$ is then defined as the closure of the image of η in $X \times G_r^n$ and the Nash modification $p : \tilde{X} \longrightarrow X$ is the first projection. By a result of Nobile [8], p can locally be formulated as a blowing up at a center $J \subset \mathcal{O}_X$ where J is generated[6] by appropriate elements g_β of the ideal of $n-r$ minors of the Jacobian matrix $(\frac{\partial f_i}{\partial x_j})_{1 \leq i \leq m, 1 \leq j \leq n}$. More precisely, for each irreducible component X_i of X, we can find

- an $(n-r) \times n$ submatrix of this matrix of which at least one $n-r$ minor does not vanish on X_i (The minors of this submatrix will be denoted by $M_{i,\beta}$ where β indicates the columns involved in this particular minor)
- a global section $0 \neq h_i \in \Gamma(X, \mathcal{O}_X)$ vanishing along all other components X_j, $1 \leq j \leq d$, $i \neq j$.

The generators of J are then

$$g_\beta = \sum_{i=1}^{d} h_i M_{i,\beta}$$

where β runs through all $n-r$ tuples of column indices of the Jacobian matrix.

In the case of a complete intersection, the Jacobian matrix does not have more than $n-r$ rows, thus making the row selection and the h_i in the above construction unnecessary and implying that J is just the ideal of the singular locus.

```
// As a first example we consider a complete intersection:
ring r=0,(t,x,y,z,a(1..5)),(dp(1),dp);
                              // A^3 plus additional variables
```

[6]Under the above simplifications of $X \subset \mathbb{A}^k$ and $I(X) = \langle f_1, \ldots, f_m \rangle$

```
                      // for blowing up
ideal I1=x^2-y^2-z^4,yz;
                      // 2 lines V(x+y,z),V(x-y,z)
                      // 2 parabolas V(x-z^2,y),
                      //             V(x+z^2,y)
                      // all meeting in (0,0,0)

// Center of blowing up is singular locus
LIB"sing.lib";        // slocus is in sing.lib
ideal sL=mstd(slocus(I1))[2];
                      // minimal number of generators
                      // of ideal of singular locus
size(sL);
--> 5
ideal blow1=I1,a(1)-t*sL[1],a(2)-t*sL[2],a(3)-t*sL[3],
               a(4)-t*sL[4],a(5)-t*sL[5];
ideal Elim1=eliminate(blow1,t);
                      // do the blowing up

// Now we would like to check that we have indeed
// 2 single and a double point in the preimage
// of V(x,y,z)
LIB"primdec.lib";
primdecGTZ(Elim1+ideal(x,y,z));
--> [1]:              // double point
-->   [1]:            // the component
-->      _[1]=a(5)^2
-->      _[2]=a(3)
-->      _[3]=a(4)
-->      _[4]=a(1)
-->      _[5]=z
-->      _[6]=y
-->      _[7]=x
-->   [2]:            // its radical
-->       ....        // output omitted
```

```
-->[2]:                 // single point
-->   [1]:              // the component
-->      _[1]=a(3)-a(5)
-->      _[2]=a(2)
-->      _[3]=a(4)
-->      _[4]=a(1)
-->      _[5]=z
-->      _[6]=y
-->      _[7]=x
-->   [2]:             // its radical
-->      ....          // output omitted

-->[3]:                // single point
-->   [1]:             // the component
-->      _[1]=a(3)+a(5)
-->      _[2]=a(2)
-->      _[3]=a(4)
-->      _[4]=a(1)
-->      _[5]=z
-->      _[6]=y
-->      _[7]=x
-->   [2]:             // its radical
-->      ....          // output omitted

// As a second example, we determine the center
// in the non-complete-intersection case:
ring r=0,(x,y,z),dp;
                        // A^3
ideal I2=xz,yz,x^2-y^4;
                        // 1 line V(x,y)
                        // 2 parabolas V(x-y^2,z)
                        //         and V(x+y^2,z)
                        // all meeting in (0,0,0)
list comps=minAssGTZ(I2);
```

```
                       // minimal associated primes
                       // of our ideal --
                       // coincides here obviously
                       // with prim. decomp.
matrix M[3][3]=diff(I2,x),diff(I2,y),diff(I2,z);
print(M);              // Jacobian matrix of I2
--> z,0,2*x,
--> 0,z,-4*y^3,
--> x,y,0
// To determine the appropriate generators
// of our center, we need to construct the
// g_beta=\sum h_i M_beta,i
// Step 1: define the three submatrices
//         and their respective ideals of minors:
matrix M12[2][3]=M[1,1..3],M[2,1..3];
matrix M13[2][3]=M[1,1..3],M[3,1..3];
matrix M23[2][3]=M[2,1..3],M[3,1..3];
ideal min12=minor(M12,2);
ideal min13=minor(M13,2);
ideal min23=minor(M23,2);
// Step 2: check for each component, which minors
//         do not vanish along the component
size(reduce(min12,std(comps[1])));
--> 0                  // all minors of M12 vanish
                       // along first component
size(reduce(min12,std(comps[2])));
--> 0                  // as before
size(reduce(min12,std(comps[3])));
--> 1                  // this is the good component
/* Important Aside:
   The numbering of the components in the output
   of minAssGTZ resp. primdecGTZ is not fixed and
   often changes when recomputing the decomp. */
...
// ... repeating these steps for the other ideals of
```

```
//     minors, we obtain:
//     comp1: M13 or M23
//     comp2: M13 or M23
//     comp3: M12

// Step 3: determine the h_i:
// check which generators of intersection of compi and
// compj does not vanish identically on compk
ideal inter12=intersect(comps[1],comps[2]);
reduce(inter12,std(comps[3]));
                       // study comp1 \cap comp2
                       // and comp3 ==> h3
--> _[1]=z
--> _[2]=0
--> _[3]=0
poly h3=inter12[1];    // inter12[1] does not
                       // vanish identically on comp1
...
// ... repeating these steps for the other two
//     components, we obtain:
//     h1 = y^2-x
//     h2 = y^2+x
//     h3 = z

// Step 4: combine information to obtain the center:
ideal center=(h1 * min13) + (h2 * min13) + (h3 * min12);
center;
--> center[1]=z^3
--> center[2]=x*z^2
--> center[3]=x*y*z
--> center[4]=x^2*y
--> center[5]=x^3
--> center[6]=y^3*z-x*y*z
--> center[7]=x*y^3-x^2*y
```

```
// Blowing up with this center now provides the
// desired Nash modification.
```

References

[1] Bravo,A., Encinas,S., Villamayor,O.:*A Simplified Proof of Desingularisation and Applications*, Rev. Math. Iberoamericana 21 (2005), 349–458.

[2] Decker,W., Lossen,C.: *Computing in Algebraic Geometry - A quick start using* SINGULAR, Algorithms and Computation in Mathematics 16, Springer Verlag (2006).

[3] Encinas,S., Hauser,H.: *Strong resolution of singularities in characteristic zero*, Comment. Math. Helv. 77 (2002), 821–845.

[4] Greuel,G.-M., Pfister,G.: *A* SINGULAR *Introduction to Commutative Algebra*, Springer (2002).

[5] Greuel,G.-M., Pfister,G., Schönemann,H.: SINGULAR 3.0, `http://www.singular.uni-kl.de/`.

[6] Hartshorne,R.: *Algebraic Geometry*, Springer (1977).

[7] Kollar,J., Mori,Sh.: *Birational geometry of algebraic varieties*, Cambridge Univ. Press (1998).

[8] Nobile,A.: *Some Properties of the Nash Blowing-Up*, Pac. J. Math. 60 (1975), 297–306.

[9] Shafarevich,I.: *Basic Algebraic Geometry*, Springer (1977).

Anne Frühbis–Krüger,
Received January 9, 2008

E–mail: *anne@math.uni – hannover.de*