

Simulation Data Management from CAD to Results with LoCo and CAViT for Large Scale LS-DYNA[®] LEGO[®] Crash Models

LEGO[®] is a trademark of the LEGO Group of companies which does not sponsor, authorize or endorse these investigations.

Thorsten Gerlinger¹, David Koch¹, Andre Haufe¹, Nils Karajan²,
Thomas Weckesser², Pierre Glay³, Alexandru Saharnean⁴, Marko Thiele⁴
¹DYNAmore GmbH, ²DYNAmore Corporation, ³DYNAmore France SAS, ⁴SCALE GmbH

Abstract

Given that in our professional lives we are dealing with highly sophisticated crash models on a daily basis, it seems obvious that we instantly thought we should be able to simulate a crash of a LEGO[®] Porsche Technic Model using the LS-DYNA FEM solver after seeing a video of a physical crash of this model on YouTube.

Setting up a process, which involves every aspect of working with CAD data, meshing, dealing with solver files, submitting and monitoring the simulations, and finally handling the result files of simulations, is an important step when developing a Simulation Data Management (SDM) system such as LoCo and CAViT. Therefore we decided to use this LEGO[®] crash as a challenge and benchmark for our software.

The real LEGO[®] models are often assembled with thousands of bricks. Handling so many parts in a SDM system on one hand and maintaining the ability to work on such models in a collaborative way with multiple users on the other is quite challenging. Initially, we set up the whole simulation process for the Porsche which is composed of 2704 individual bricks. But when we showed the results of the LEGO[®] Porsche crash simulation to the c't magazine (a widely read German computer magazine) and ADAC (General German Automobile Club) who had performed the initial physical crash test in 2017, they suggested doing another LEGO[®] crash scenario. This time a LEGO[®] Porsche was supposed to crash into a LEGO[®] Bugatti model at 60km/h and a LS-DYNA simulation should predict the outcome of the crash before the physical test was going to be conducted. The LEGO[®] set number 42083 of the Bugatti Chiron is even bigger than the Porsche model and consists of 3599 bricks. The results of the simulation were then evaluated and presented to ADAC and c't magazine to provide our prediction of the upcoming real physical crash.

Later, the comparison of our prediction with the real crash results revealed that many details have been predicted correctly by the simulation.

The final LS-DYNA model of both car models consisted of more than 45 million elements. Preprocessing as well as getting the model to run on an HPC system and handling the few but large result files has proven to be challenging in many ways.



Introduction

Playing with LEGO® bricks is something many engineers might have enjoyed during their childhood. Building any kind of mechanical construction allows creativity and complexity to an extent which probably contributed to their fascination and finally in their decision of becoming engineers. It's interesting to see how many of them are still fascinated by LEGO® even in their adult life. Especially for children, or for those with an active inner child, crashing these models into each other is even more fun, because seeing all those bricks fly all over the place is just fascinating, beyond any scientific or professional aspect.

Given that in our professional lives we are dealing on a daily basis with highly sophisticated crash models, it seems obvious that, when seeing videos such as this real crash of a Porsche 911 GT3 RS LEGO® Technic (42056) model conducted at 46km/h with an 40% offset barrier on YouTube^{[2][4]}, we instantly think we should be able to simulate this with the LS-DYNA^[5] FEM solver.

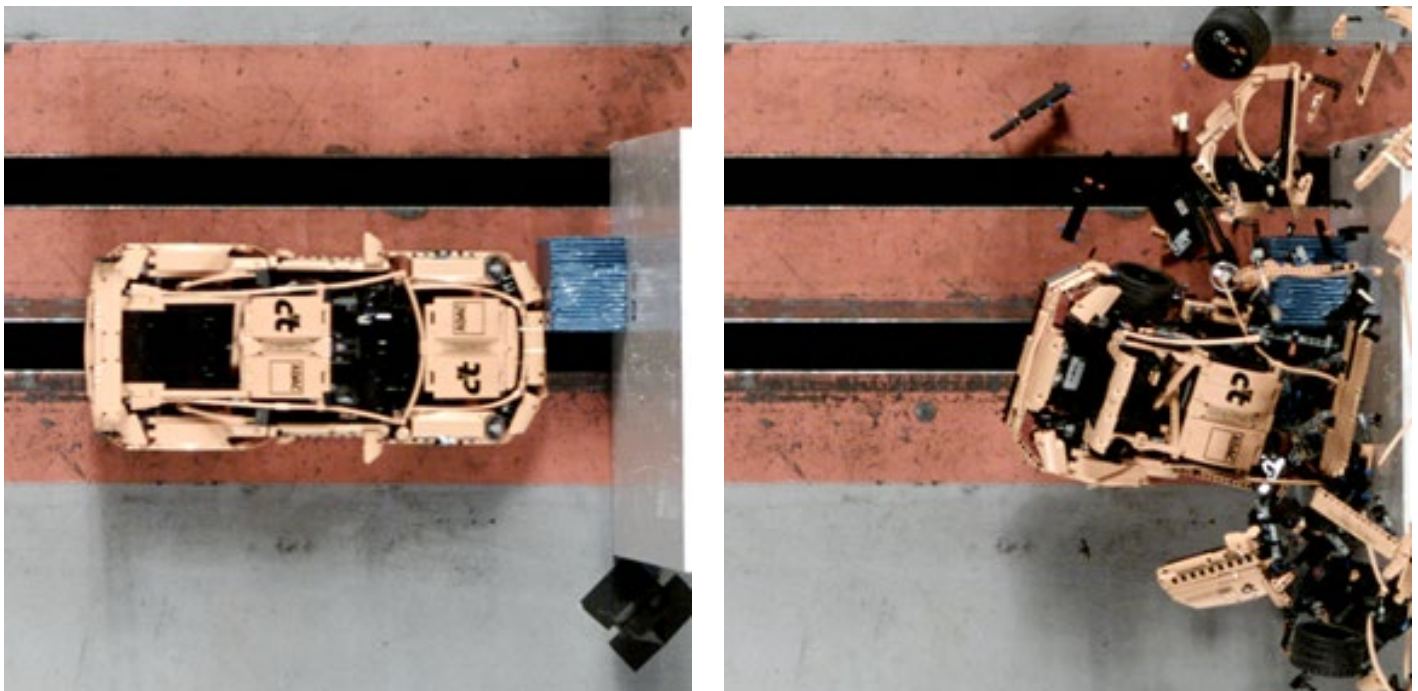


Fig. 1: Physical crash of Porsche 911 GT3 RS LEGO® Technic (42056) model conducted at 46km/h with an 40% offset barrier by c't and ADAC^{[2][4]}.

Besides from the desire to do so, just because we can, there are also several reasons why it's actually worth doing it. Setting up a process, which involves every aspect of working with CAD data, meshing, dealing with solver files, submitting and monitoring the simulations, and finally handling the sparse result files of simulations, is an important step when developing a Simulation Data Management (SDM) system such as LoCo and CAViT^[6]. The real LEGO® models are often assembled with thousands of bricks. Handling so many parts in an SDM system on one hand and maintaining the ability to work on such models in a collaborative way with multiple users on the other is quite challenging. Just opening and working with CAD data of so many parts in modern CAE preprocessing tools is a resource demanding and complex task. And beyond that, a number of other issues also have to be solved, for example reusing already meshed bricks in the overall model or even in different models to avoid redundant work. These all are also common issues that have to be dealt with in real car projects.

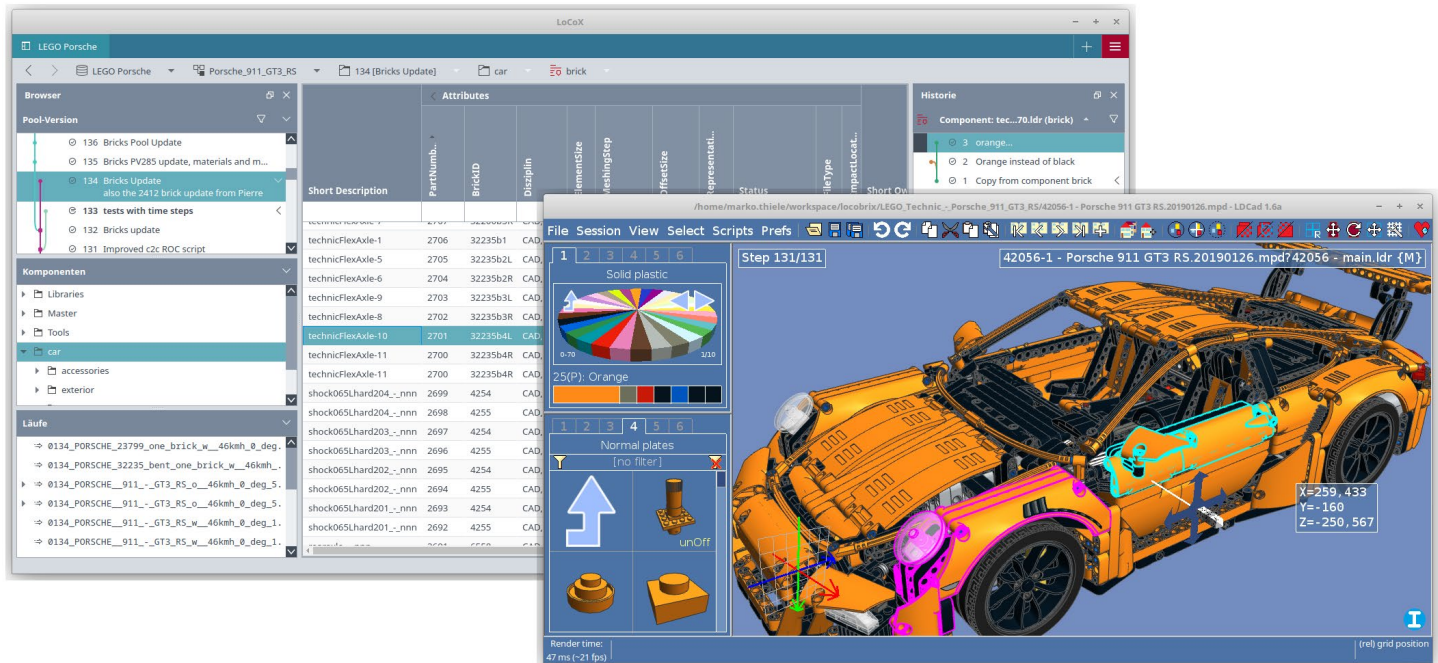


Fig. 2: Working with CAD within LoCo.

Another aspect is that using LEGO® for teaching about the setup of simulation processes configured with LoCo or mechanical engineering and FEA might be quite motivating. It could help class attendees of our training classes or young students to stay focused and pick up some SDM and FEA ideas, see the fun in physics and the challenges for team work in case of real and complex product development.

When it came to actually approaching the goal to set up a simulation process for generating LEGO® models made out of thousands of bricks it became clear that this requires teamwork. The whole process was setup with LoCo by colleagues from SCALE GmbH in the offices of Ingolstadt and Dresden in such a way that it would be relatively easy to import LDraw^[7] models and use the positioning information of the bricks in order to position the actual meshed bricks. This way each brick would have to be meshed only once and could be used instantly wherever it appeared in the model. The bricks are also maintained in a library within the SDM system such that it can be used throughout various different models and the actual meshing process could be distributed over various offices of DYNAmore and SCALE.

Working with Digital LEGO® models

Searching the internet, one can find a seemingly endless supply of resources, including pictures, videos, construction manuals and 3D CAD models on sites such as ldraw.org^[7] or bricklink.com^[8]. These 3D models are commonly made with special CAD software, like [LDCad](#)^[9], [LeoCAD](#)^[10], [Studio](#)^[11] and complementary tools such as [LDView](#)^[13]. Furthermore, there is a vivid community at ldraw.org^[7], maintaining a library of CAD Data for all the numerous bricks that are available from LEGO®. This variety of source material also implies an advantage for using LEGO® focused CAD programs.

In order to get started with creating own CAD models, the aforementioned software Studio from BrickLink might be a good choice, since its easy to understand concepts allow even young students of age 10+ to build simple models.



Fig. 3: Model opened in Studio LEGO® CAD software.



Fig. 4: Photorealistic rendering generated with Studio.



Fig. 5: Photo of physical build of scale car model.

Before actually starting with the sophisticated Porsche 911 GT3 RS LEGO® Technic (42056) model, henceforth referred to as Porsche model, a model of a smaller car was created. This gave us the possibility to get used to the tools and the setup of the processes needed later for working with the considerably larger Porsche model.

The model shown in Fig. 3 and Fig. 4 was created by us and is now publicly available and can be obtained from bricklink.com^[8]. So if you want to take a look, just go ahead and try yourself. On BrickLink, as well as on the [OMR \(Official Model Repository\)](#)^[12] at ldraw.org^[7] you will also find many other models of the same kind, just to investigate and play with.

Furthermore, the online store at the BrickLink website offers LEGO® bricks for purchase. This makes it quite convenient to build a physical model, which was previously created with the software. Fig. 5 shows the physical build of our starting model.

For more advanced models, like the Porsche model that will be shown henceforth, or for those favoring open source, LDCad might be the preferred choice because it offers a significantly better performance and handling of flexible parts. Another good tool is LeoCAD which is also open source and has a somewhat easier interface so it might be suited for the first steps and experiments with LEGO® CAD software. LeoCAD as well as LDCad are also better suited for integration with LoCo since they feature an easy command line interface to open models for editing.

Setting up the Simulation Process with the Simulation Data Management System LoCo

Aspects of the teamwork involved

When the idea took off to set up a simulation process for generating LEGO[®] models made out of thousands of bricks, it became clear that this ambitious task would require teamwork. Therefore, the whole process was set up using LoCo by colleagues from SCALE GmbH in the offices of Ingolstadt and Dresden, Germany. LoCo is a SDM system that allows to set up simulation processes and to handle the data required for simulation and sharing that data among various team members.

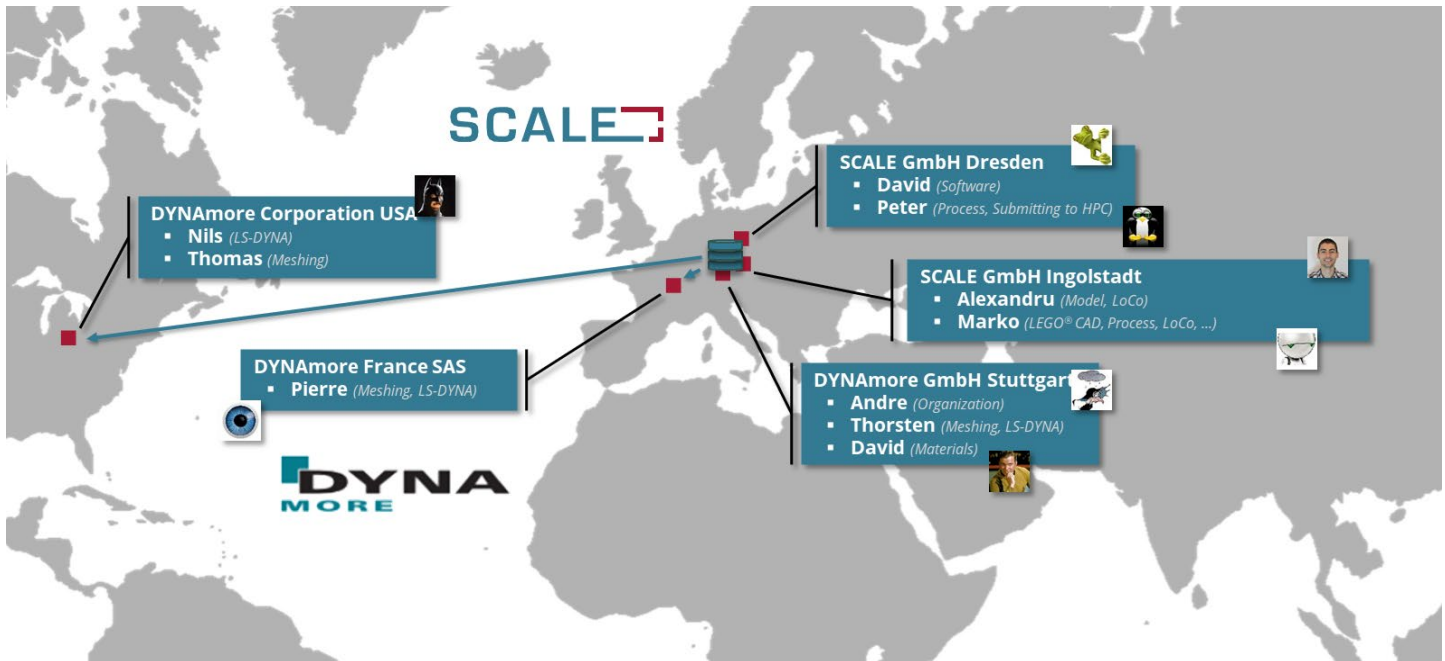


Fig. 6: Tasks during collaborative effort to set up the simulation processes with the LoCo SDM system.

The main challenge when working together on the same model is that no more than one person can work on the same pieces of data at the same time. There are several approaches to deal with this problem. The most user friendly approach is the realization of a system that allows for real-time editing among several team members within the same files. This has been done for simple text editors in projects such as Etherpad^[14] and lately also for office documents, e.g. Google Docs^[15] or Collabora Online^[16]. However, the typical CAD and CAE preprocessing tools are not capable of such simultaneous editing. In order to collaborate on the same CAD or CAE model, a commonly used approach is to split up the model into several files. As the size of the team grows, the importance of splitting up the model into ever smaller parts increases. This is also something that is seen in productive examples of real car projects. If consequently applied, this approach results in setups where each part in a whole car assembly is held in an individual file. For complete car projects the amount of parts/files to be handled becomes quite large such that the SDM system has to be able to deal with several thousands of parts/files at the same time. This is especially challenging with regard to the user interface as well as the underlying mechanisms such as synchronization of data. Therefore, creating such a “worst case” scenario is important for the development of an SDM system and was part of the main motivation for creating the LEGO[®] examples.

Process setup

The most used format to describe LEGO® models is LDraw^[7]. In this format basically every line is just providing the transformation information, a color and a reference to one brick in the standardized LDraw brick library. When trying to go from the LDraw format to actual LS-DYNA simulation models it seemed natural to use a similar approach as used in the LDraw format. The goal is to be able to import LDraw models relatively easy and use the positional information of the bricks in order to position the actual meshed bricks. This way each brick would have to be meshed only once and can be used instantly wherever it appears in the model. The idea is, that, if a library of meshed bricks in the same positions as the bricks in the LDraw library is available, it should be easy to create simulation models for any of the countless LDraw models available online. Maybe at some point even easy enough for young students. A created mesh of one brick can be used over and over again by utilizing ***INCLUDE_TRANSFORM** cards of LS-DYNA to import the brick to its various locations. Within LoCo the meshed bricks are also maintained in a shared library pool that can be used throughout various different projects. This makes it easy to share the same version of the meshed bricks throughout all projects.

CAD Data

The CAD Data for the Porsche model^[17] has been obtained as a LDraw file from the OMR (Official Model Repository)^[12]. It was published by Philippe Hurbain and released under the CCAL version 2.0^[18] which allows every kind of usage. Within the LDraw file a typical line contains the transformation and color information as well as a unique brick ID for one brick of the model.

```
0 FILE 42056 - main.ldr
0 main
0 Name: 42056 - main.ldr
0 Author: Philippe Hurbain [Philo]
0 !LDRW_ORG Model
0 !LICENSE Redistributable under CCAL version 2.0 : see CReadme.txt

0 !THEME Technic

0 ROTATION CENTER 0 0 0 1 "Custom"
0 ROTATION CONFIG 0 0
1 71 -0.567 0 -180.567 0 0 1 0 1 0 -1 0 0 64179.dat
1 1 -60.567 0 -160.567 -1 0 0 0 0 -1 0 -1 0 6558.dat
1 1 -60.567 0 -200.567 -1 0 0 0 0 -1 0 -1 0 6558.dat
1 0 -40.567 -40 -180.567 0 1 0 0 0 1 1 0 0 60484.dat
1 0 -40.567 0 -230.567 0 1 0 0 0 1 1 0 0 2780.dat
1 0 39.433 0 -230.567 0 1 0 0 0 1 1 0 0 2780.dat
```

Fig. 7: First lines of the LDraw model file of the Porsche model.

The files can also contain nested structures where parts of the model are referenced and transformed as a whole. This makes working with part assemblies, for example a wheel assembly, easier as they can be built once and placed on various positions in the model.

For importing these structures into LoCo, a script was implemented, which splits the LDraw file into individual files for each brick. Afterwards, the individual files were imported in a batch operation and also, the nested structure for subassemblies was rebuilt within LoCo. This allows some team members to work on one subassembly while others work on another one. This becomes especially handy when using the LiveMode feature of LoCo^[3]. Here the SDM system keeps track of who is working on which parts of the model, such that users do not need to merge the versions of the model parts they were working on later in time.

Once the CAD data was imported from the LDraw file to LoCo, the users are able to work directly with e.g. LDCad within LoCo as shown in Fig. 2. For this purpose a small wrapper script was written, which combines the files with single lines per brick into one big file and opens it in LDCad or LeoCAD. After saving the changed assembly in LDCad, the file is split again and only the changed and new bricks are saved back to LoCo. The user does not notice this process. He just clicks on an assembly and requests it to be opened in LDCad. New versions of all files and folders are created and synchronized automatically to all other team members working on the project. In this case LoCo also acts as a collaboration platform, giving the possibility to try out different versions of the actual LEGO[®] model or create a model in cooperation with others.

Meshing

The main collaborative effort was to do the actual meshing. This has been done by colleagues from DYNAMore GmbH, DYNAMore Corporation in USA, DYNAMore France SAS and SCALE GmbH using LoCo to organize and exchange the meshed bricks (Fig. 6, Fig. 8). Each brick was initially meshed as a 2 mm as well as a 1 mm variant which are kept together in LoCo in one place such that one is able to easily generate simulations using either of the mesh resolutions. Together with the LS-DYNA models for each brick, the CAD data and cleaned geometry were also stored in LoCo as ANSA^[24] DB files.

The original CAD data for each brick from the LDraw brick library is most often available in the form of a simple triangulation representing the surface of the brick. This can easily be converted to *.stl or *.obj files using LeoCAD or LDView and imported to ANSA or Hypermesh. However, these meshes are not suited at all for conducting simulations. Initial experiments trying to use the simple original STL meshes as rigid bodies have not been very promising and therefore it was decided to create primarily tetrahedral volume meshes with a target edge length of 1 mm and 2 mm.

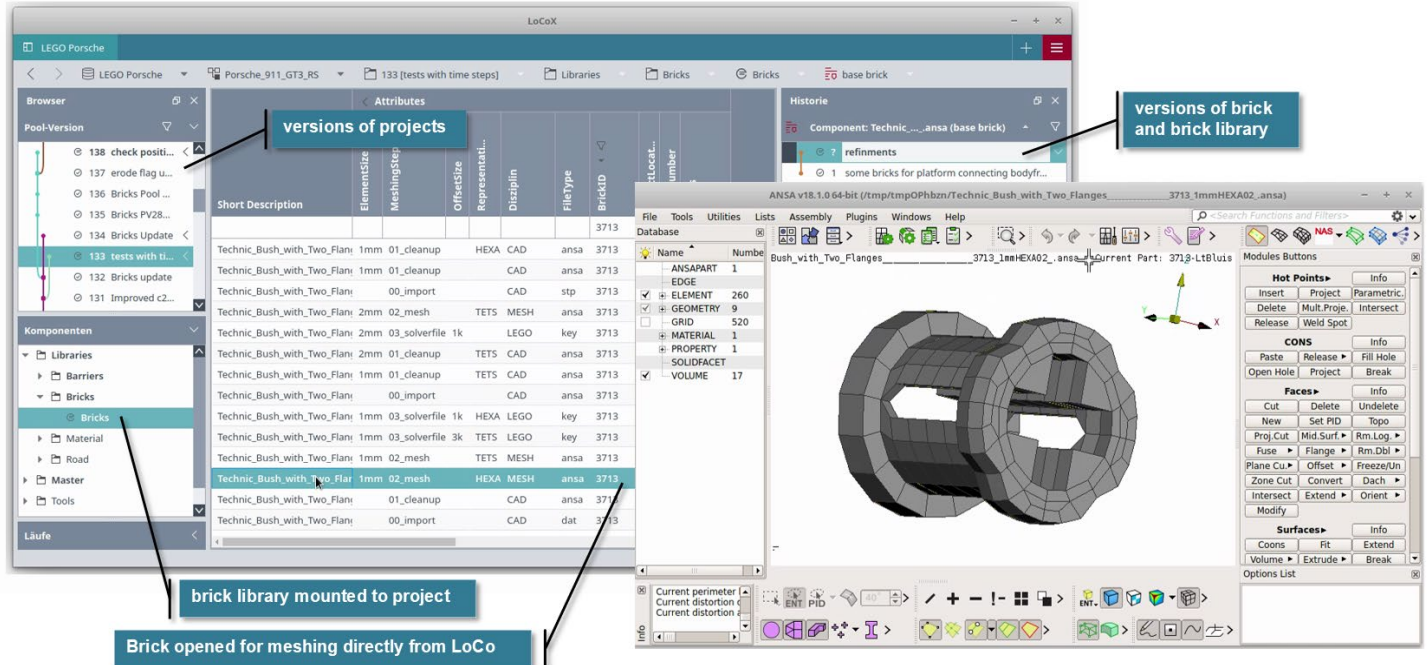


Fig. 8: Managing the library of meshed bricks in LoCo

Even though the *.stl files could be used as a basis for creating the required FEM meshes, they are actually not very convenient to work with in meshing tools. This is because they are not made out of real primitives and therefore already contain an inaccuracy, since they are composed of polygons instead of native geometries.

Fortunately, the Porsche model has also been available as “real” CAD data on GRABCAD^[19] where the user dk^[20] has kindly allowed us to use the geometric data of the bricks for our investigations. The geometry from GRABCAD only had to be positioned coincidental to the geometry of the LDraw geometry and could then be used for the subsequent meshing step.

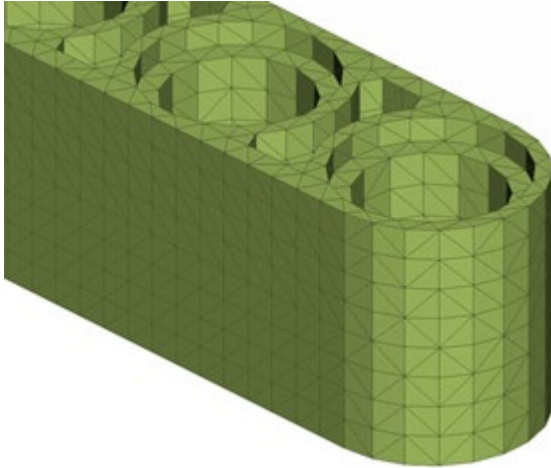


Fig. 9: Technic Beam 13, BrickID 41239 1mm Tetra mesh, 33123 elements

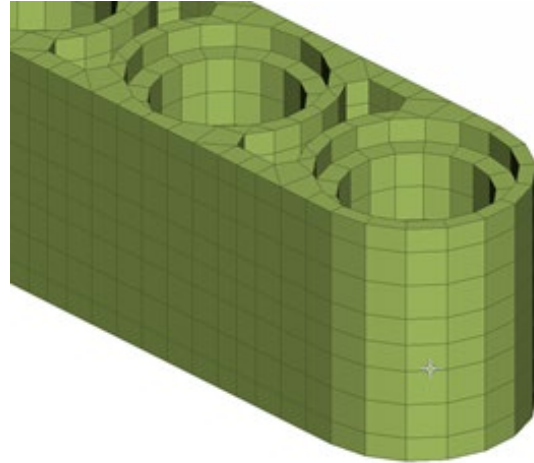


Fig. 10: Technic Beam 13, BrickID 41239 1mm Hexa mesh, 5124 elements

One drawback when using volume elements is that this results in models with many elements. This can be improved by using hexahedral elements. As shown in Fig. 9 and Fig. 10 this can dramatically reduce the number of elements. For the final model, where bricks such as #41239 are used 29 times throughout the whole model, this saves more than 800,000 elements for the final simulation.

The resulting complete Porsche model contained more than 2700 bricks, each handled as an individual part. For the variant using the bricks meshed with 1 mm target element size this model contained an astonishing amount of roughly 19.5 million elements. This exceeds the usual model size for integrated simulations of full vehicle crashes used in real virtual product development at OEMs by a factor greater than two. However, using the 2 mm variant results in loss of many details of the bricks since the contact situation, i.e. the way how the bricks stick to each other, is crucial for the overall model behavior. Hence, it seems advisable to use at least the 1 mm variant. The average run time on a high performance cluster (HPC) system using 192 CPUs for the simulation was 22 hours for 120 ms of simulated time.

Assembly

Other files needed for the final simulation, such as a master file with all the control cards, barrier models, guiding blocks (that had been attached below the vehicle to make sure it would run straight), files with material cards and so on are also maintained in LoCo. Finally, a script is stored together with all the components of the model, which is executed by LoCo when assembling an actual LS-DYNA solver deck. This script takes the transformation information of each brick and translates them to LS-DYNA ***INCLUDE_TRANSFORM** and ***DEFINE_TRANSFORMATION** cards (Fig. 11) such that each ***.key** file of the bricks is imported over and over again.

The same script also creates session files for LS-PrePost^{®[21]}, Animator^[22] and META^[23] for correct coloring when visualizing the resulting crash animations during post-processing.


```

#####
$ Include - Transform for:
$ dashboard2_-_nnn_____558_3070b_000000_----_3b16a7a2.ldr
#####
$
*DEFINE_TRANSFORMATION
  20115001
$
$  Rotation:
$
$   ROTATE      1.0      0.0      0.0      0.0      0.0      0.0      90.0
$
$ um X
$   ROTATE      1.0      0.0      0.0      0.0      0.0      0.0      -90
$ um Y
$   ROTATE      0.0      1.0      0.0      0.0      0.0      0.0      -0
$ um Z
$   ROTATE      0.0      0.0      1.0      0.0      0.0      0.0      180
$
$  Translation:
$      x      y      z
$   TRANSL  -0.2268      -80  -109.827
$
$ final rotations
$
$   ROTATE      1.0      0.0      0.0      0.0      0.0      0.0      -90.0
$   ROTATE      0.0      0.0      1.0      0.0      0.0      0.0      -90.0
$
$
*INCLUDE_TRANSFORM
Tile_1_x_1_with_Groove_____3070b_1mm03_.key
$# idnoff  ideoff  idpoff  idmoff  idsoff  idfoff  iddoff
   20115000  20115000  20115000      0  20115000  20115000  20115000
$# idroff
   20115000
$# fctmas  fcttim  fctlen  fcttem  incout
$# tranid
   20115001
$

```

Fig. 11: Automatically generated solver cards for including bricks with transformation.

With this setup in place it becomes easy to apply changes to the LEGO® CAD model by using e.g. LDCad or importing other LEGO® models and run these as new simulations. Provided the meshed bricks already exist for the bricks used in the model, the simulations can be run right away.

Setup of different load cases and model variants

Once the entire simulation process has been set up and a running simulation model has been created it becomes easy to set up different load cases using LoCo. Besides the 40% ODB offset barrier crash shown on [YouTube^{\[2\]\[4\]}](#), load cases for frontal rigid wall, 50% and 25% offset barrier, crashes with 30° impact angle and various different velocities were set up as well.

LoCo has the ability to use attributes defined on components and parameter values in order to relate them automatically to specific simulations. This makes it possible to create a high amount of different variations on one virtual product without having to relate all components (files, includes) individually to each load case. Therefore a change applied to e.g. one brick is automatically taken over to all defined use cases and submitting the jobs for many load cases to evaluate an overall status of a simulation project becomes effortless.

A car to car crash simulation using the Multi-Stage-Assembly feature of LoCo was also set up. With this feature one can use the result of one assembly, e.g. the assembled Porsche model, within another assembly for a different simulation.

LS-DYNA Modeling

Global Contact

Thanks to [LS-DYNA^{\[5\]}](#) contact algorithms, the strategy of the contact is relatively straightforward for users in automotive crashworthiness applications. Only one single surface contact is needed to take care of the full vehicle, including the ODB offset barrier. This has the huge advantage to eliminate the need to know contacting surfaces. ***CONTACT_AUTOMATIC_SINGLE_SURFACE** has been used for this purpose together with the option **SOFT=2** to achieve a segment-based contact detection. This provides an efficient way of treating interaction between disjoint elements and such that it prevents penetration between parts or self-penetration of the same part. In order to include all the bricks into the contact surface a ***SET_PART_COLLECT_TITLE** card has been defined for each brick.

```
$
*SET_PART_COLLECT_TITLE
global contact
    1001      0.0      0.0      0.0      0.0
    2
$
```

Fig. 12: Definition for the set defining the contact surface.

This is the identical card in each brick. And since the bricks are imported multiple times with offsets for the individual part IDs all the parts for the overall simulation are collected automatically depending on which bricks are actually used in the models. The big advantage of this approach is that one does not have to define a contact surface prior to the simulation but instead the contact surface depends on which bricks are included at the time of assembling the model. Therefore, it becomes particularly easy to interchange bricks, introduce new bricks or delete undesired bricks without having to manually update the contact surface.

Material definition of the LEGO® bricks

In order to be able to reproduce realistic part deformations and material behavior, tensile samples were taken from LEGO® components and tested at the DYNAmore materials test laboratory in Stuttgart, Germany. Due to its large flat surface, brick #59349 is very well suited for this purpose. The specimens were then investigated quasi-static in a tensile testing machine, accompanied by a digital image correlation (DIC) measuring system to capture the local straining of the samples. Finally, the ***MAT_024** material card (elasto-plastic material model with isotropic hardening) was calibrated using a reverse engineering strategy within a state-of-the-art industry accepted process.

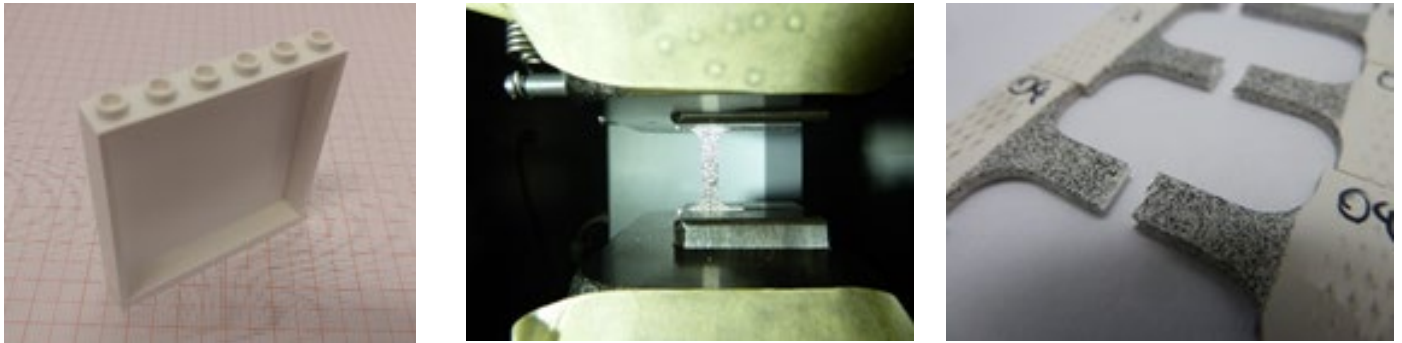


Fig. 13: Material testing to calibrate LS-DYNA MAT24 cards.

Since the Porsche model contains also a few rubber parts, generic data collected from public sources has been used to feed ***MAT_077_0** (general hyperelastic rubber model) in the final model.

Clamping force

The bricks are held together by squeezing the studs into a slightly smaller bearing of the neighbor brick that needs to be connected, thereby exerting a clamping pressure between the stud and the inner side of the neighboring brick. The two bricks are now connected together by simple friction and can only be separated in tension when the law of dry friction is violated between the stud and the inner surface of the neighbor brick. At first sight this friction-based connection of the bricks should be straightforward to incorporate into the model, as friction is naturally included in the standard contact definition in LS-DYNA.

The first simulation results of our models looked promising but did not capture the physics too well. One could clearly see that on many occasions the bricks were separating too easily. Especially smaller non-technical bricks that are only connected through studs were behaving differently than what was observed in the physical tests. As we were challenged by c't magazin and ADAC to actually predict the outcome for a real physical test of a car to car crash, we needed to have a closer look at these friction-based connections.

The coarse discretization of the connecting studs shown in Fig. 15a reveals the reason for the wrong estimation of the brick connections in tension. With such a coarse mesh it is very likely that the diameter of the studs is simply too small which leads to no clamping force at all and thus, a separation of two bricks can be achieved without resistance. The opposite can also happen where the stud diameter is actually correctly modelled such that the clamping pressure can build up. But due to the coarse mesh, the angles between the triangular segments of the stud surface cause an additional geometric resistance to the friction when sliding out of the neighbor brick and thus, the needed separation force is way too high. A solution to this problem could be simply to use a finer mesh as shown in Fig. 15b. But this would also significantly increase the element count such that the overall model will become too expensive to solve.

This is why the clamping force has to be homogenized and modeled using a so-called tied contact that will separate the tied entities when a stress or force criterion is met. In LS-DYNA there are several different ways to achieve this behavior. The available tied contacts can either be switched inactive after the tying breaks or will act as a “normal” contact thereafter. To keep our separate global single surface definition we decided to go with the option to deactivate the tied contact after it breaks. This definition is also available in terms of a stress or force based criterion. To be mesh independent it appears that a stress based criterion is the way to go. But as the mesh is not fine enough, this led to similar problems as described above. Following this, we decided to take the tedious route and use a mesh-dependant tied contact that allows us to define a maximum normal and shear force for each node in the tied contact.

Following this, the clamping forces for the studs and also for the axles are incorporated in the model by using a ***CONTACT_TIEBREAK_NODES_ONLY** contact (Fig. 14).

```
*CONTACT_TIEBREAK_NODES_ONLY
$      cid                      title
$      ssid      msid      sstyp      mstyp      sbboxid      mboxid      spr      mpr
$      7002      1001        4         2          vdc      penchk      bt      dt
$      fs         fd         dc         vc          20.0          sfst      sfmt      fsf      vsf
$      sfs         sfm         sst         mst          -0.15      -0.15
$      nflf        sflf        nen         mes
$      1.45E-4      3.0E-5
```

Fig. 14: Keyword for the tied contact definition to achieve a clamping force.

This contact uses a slave set which contains the nodes on top of the studs of each brick (Fig. 15). The set is defined using the COLLECT option such that all nodes of all bricks are collected just as they appear in the model. The master set used in the tiebreak contact is the same part set that is used for the global contact and contains all bricks. This way whenever a node of a stud is close to the segment of a brick it will be tied to the brick. How close a node needs to be can be defined using the SST and MST parameters.

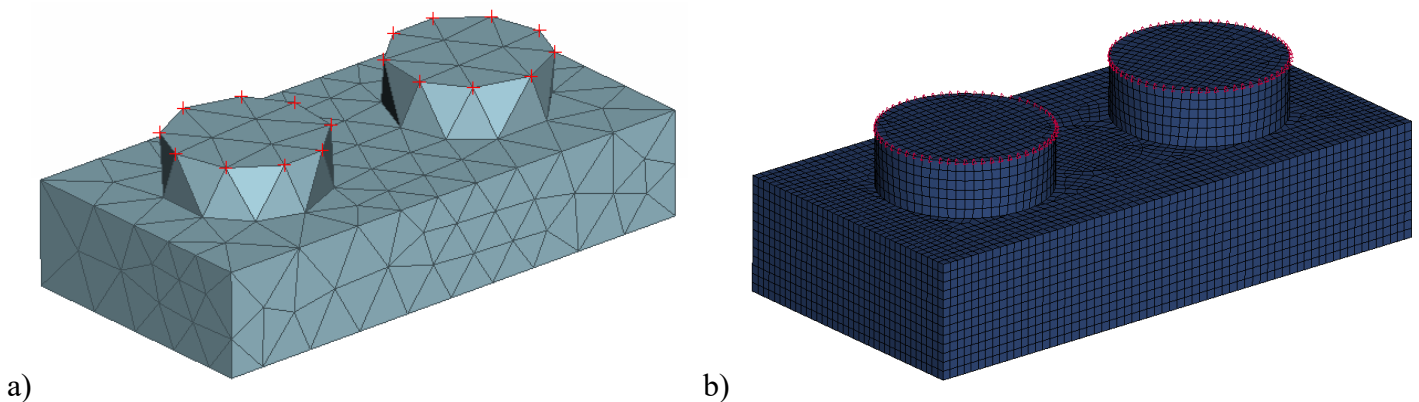


Fig. 15: Coarse (a) and fine (b) discretization of a 1x2 plate (brick #3023). Nodes highlighted in red on the studs will be part of the node set with **ID 7002** that will be tied to bricks in close proximity.

In order to determine a first estimation for the normal and shear forces at which the contact breaks, experiments have been conducted and applied to the number of nodes typically present on one stud. (Fig. 16)

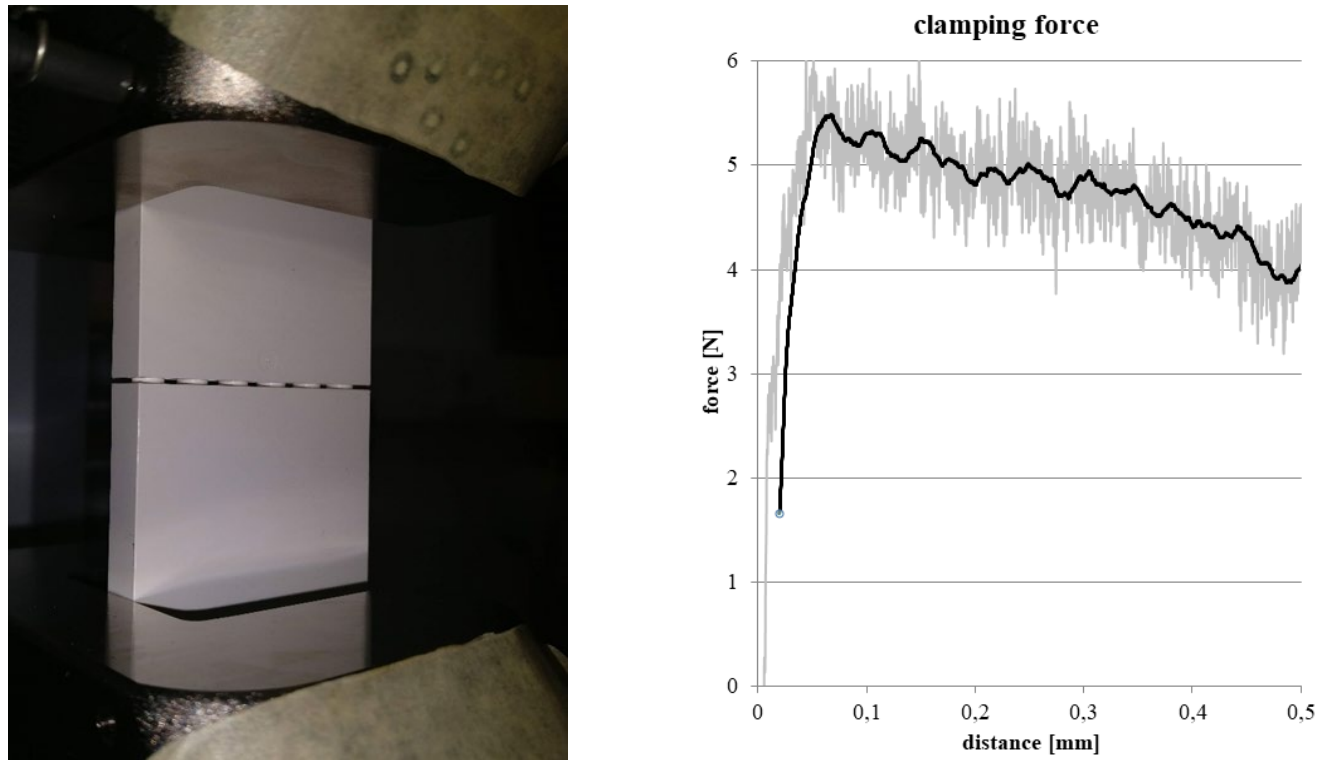


Fig. 16: Physical tests for initial estimation of clamping forces.

The measurements from this Experiment have been used to calibrate the values for NFLF and SFLF in the ***CONTACT_TIEBREAK_NODES_ONLY** contact contact card (Fig. 14) using LS-DYNA simulations (Fig. 17).

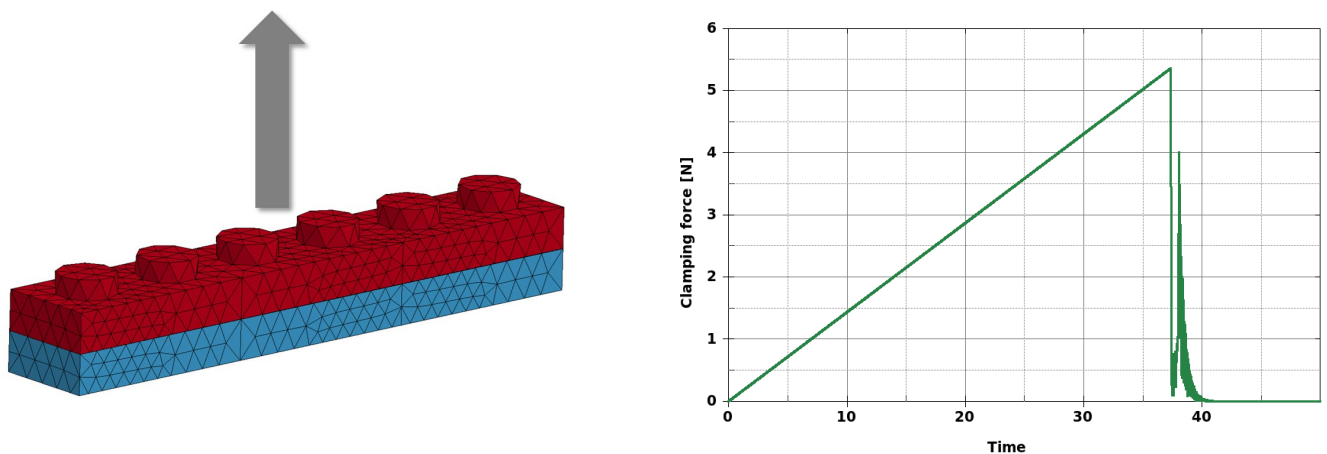


Fig. 17: Calibration of the clamping forces by simulation

The initial separation forces have later been validated and fine tuned using dynamical tests of the small scale car on a wooden crash sled. (section “Small scale car model”)

Rotation of wheels

The initial velocity of the model is defined using LoCo-parameters which then are used in a parameterized ***INITIAL_VELOCITY_GENERATION** card. In this card the initial velocities are applied to a given set of nodes. This set of nodes contains all the nodes of the model. This is achieved by defining a ***SET_NODE_LIST_GENERATE_COLLECT_TITLE** card in each key file, using the same ID for the set. In this manner this set collects all the nodes of the model.

To simulate the rotation of the wheels, one needs to define different initial velocities for the parts resembling the wheel. Hence for each wheel part or brick an ***INITIAL_VELOCITY_GENERATION** card is defined additionally in the corresponding ldr file. In this case the card contains both the translational as well as the angular velocity values. The values for the angular velocities are also freely defined by LoCo-parameters with expressions depending on the wheel radius as well as the LoCo-parameters defining the vehicle translational velocity.

In the assembly process the same script that interprets the ldr files to write the include cards into the master file also copies any following ***INITIAL_VELOCITY_GENERATION** cards from the ldr files into the master file. The ID of the card is set correspondingly to the ID offset for the entire key file.

Use as benchmark model

One further aspect in regard to the practical importance of setting up such a large model is the use as a benchmark model. With this, one could evaluate how such big models behave on HPC systems and use this experience in preparation for the time, when actual real car crashworthiness models are getting in the same range in terms of model size, memory usage and computing time.

The MPP (Message Passing Parallel) version of LS-DYNA uses a decomposition technique in subdomains in order to divide the model and the MPI protocol to communicate. Each subdomain is managed by one processor. Because just simply increasing the number of processors would not ensure to speed up the simulation significantly, particular attention was paid to finding better domain decomposition than the default one. The Fig. 18 shows the selected decomposition principle on eight processors. The model is cut into slices according to the Y-axis which ensured a better load balancing of calculation cost among processors.



Fig. 18: Domain decomposition for simulation with 8 CPUs.

Due to the large amount of include files (for the car to car crashes it where 6246 ***INCLUDE** statements), it was necessary to have an eye on the management of **SET** cards in each include. Avoiding large numbers of sets of the type **GENERATE** helped reduce the initialization time of the car to car setup to 25 minutes.

Validation

Small scale car model

As already mentioned in section “Working with Digital LEGO® models” as a predecessor of the complex Porsche model a much more simple and smaller scale car model was built with only 134 bricks. However, the first simulation results seemed unconvincing, since the car was heavily bouncing backwards. The expectation was that it gets a lot more spin when hitting the barrier with only one edge of the car. Therefore, the decision was made to construct a small wooden crash sled in order to get some means of physical validation. The process of building this sled took about a weekend’s time.



Fig. 19: Crash sled (left) and positioned car for 25% overlap frontal crash (right).

The sled is driven by a slingshot and can accelerate the car to approximately 15-20km/h which is much higher than one would estimate and made the bricks of the car fly all over the place.

The test was then filmed using the slow motion mode of two smartphones. One was placed on top of the housing to create a video showing the crash from above. The other was placed on the left side of the crash sled to provide a side view. Both cameras (Samsung S7 and Pixel 2) have been able to capture video at 240fps and produced reasonable results for such a low budget setup.

The achieved velocity in the physical test has been obtained by analyzing the frames of the video right before the crash. Since the physical dimensions of the car, the time between two frames and the amount of pixels the car moved between frames were known, it was easy to obtain the actual velocity of the car right before the crash. The determined speed of 17km/h was then used to create a simulation for validation.

Fig. 20 shows the results for the side view comparing images from the test videos and the simulation. In accordance to the simulation, the physical test also confirmed that the car is bouncing straight back from the barrier right after the impact and does not get much spin.

Two videos comparing the side view^[25] results from simulation and crash as well as the top view^[26] are also posted on YouTube®. For a simulation with such little tweaking for validation it is already impressive how well the results of the simulation and the test compare.

Having the simulation process in place and the confirmation that the obtained results could be within a reasonable range the road was paved to approach the more challenging task of creating the simulation for the Porsche model.

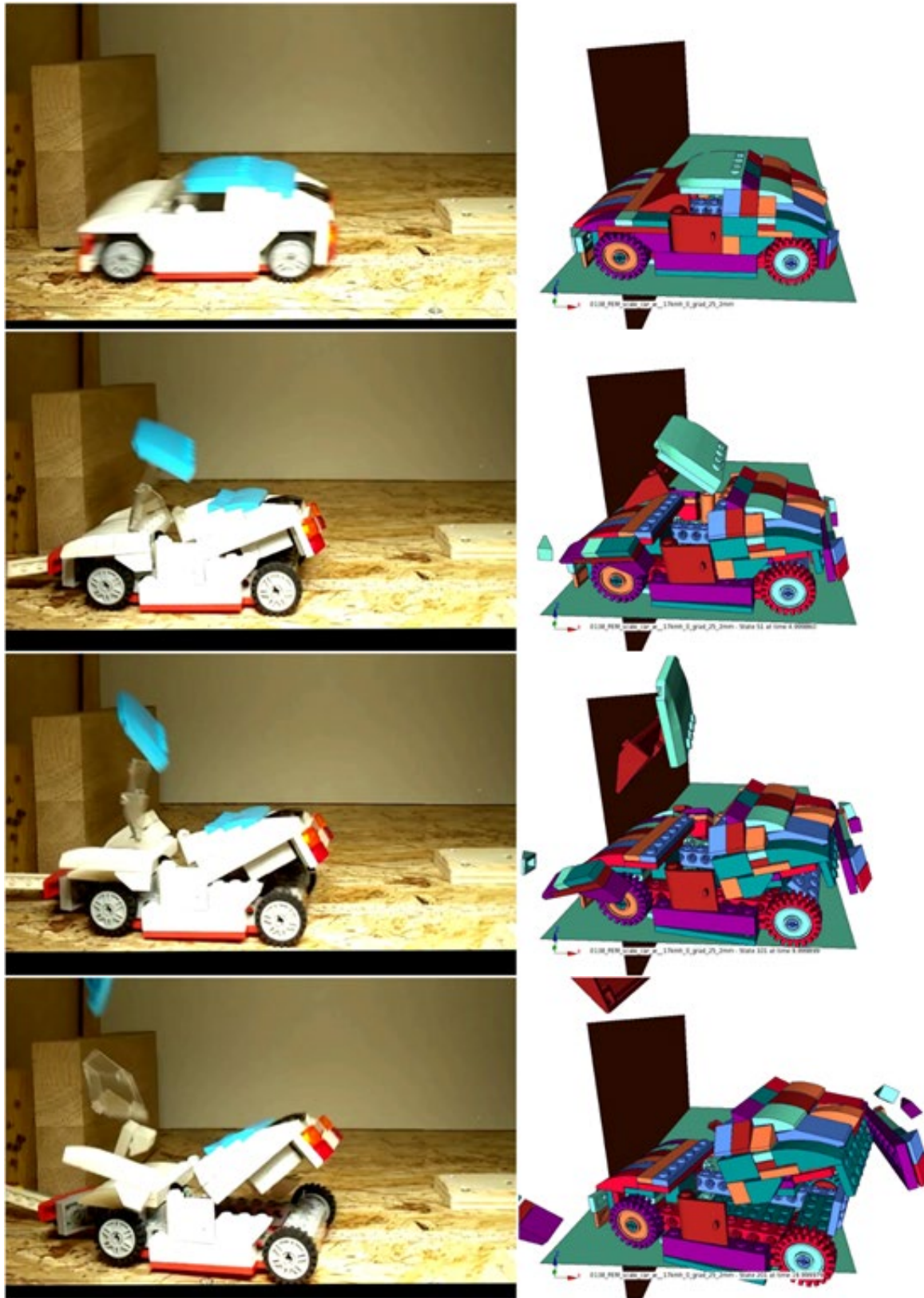


Fig. 20: Physical test compared to simulation for 25% offset crash at 0, 5, 10 and 20 ms of the scale model.

c't magazine / ADAC crash of Porsche model

When c't magazine published their challenge for the crash test^[27] of the Porsche model on YouTube, they left a big cliffhanger by initially not showing the actual crash and asking the community how, in their opinion, the model would behave during the crash. At that point many CAE colleagues instantly stated that LS-DYNA should be able to predict that behavior by simulation. Some of them also downloaded the model from GRABCAD^[19] and opened it in ANSA^[24]. However, almost immediately it became clear that one would not be able to create any simulation right in time for the c't magazine challenge.

However, the videos later posted by c't and ADAC on YouTube provided good material in order to be used as virtual validation of the simulations to come. For example, the exact positioning of the car as well as the dimensions and the positioning of the barrier have been identified from single images of the videos posted on YouTube.

Element size

Surprisingly the used target element size had a relatively big impact on model behavior. As explained in section “Meshing”, meshes were created with 2 mm and 1 mm target element size. With these different meshes, when looking at simulations which otherwise use the same parameters, one can clearly see that the behavior of the models are quite different.

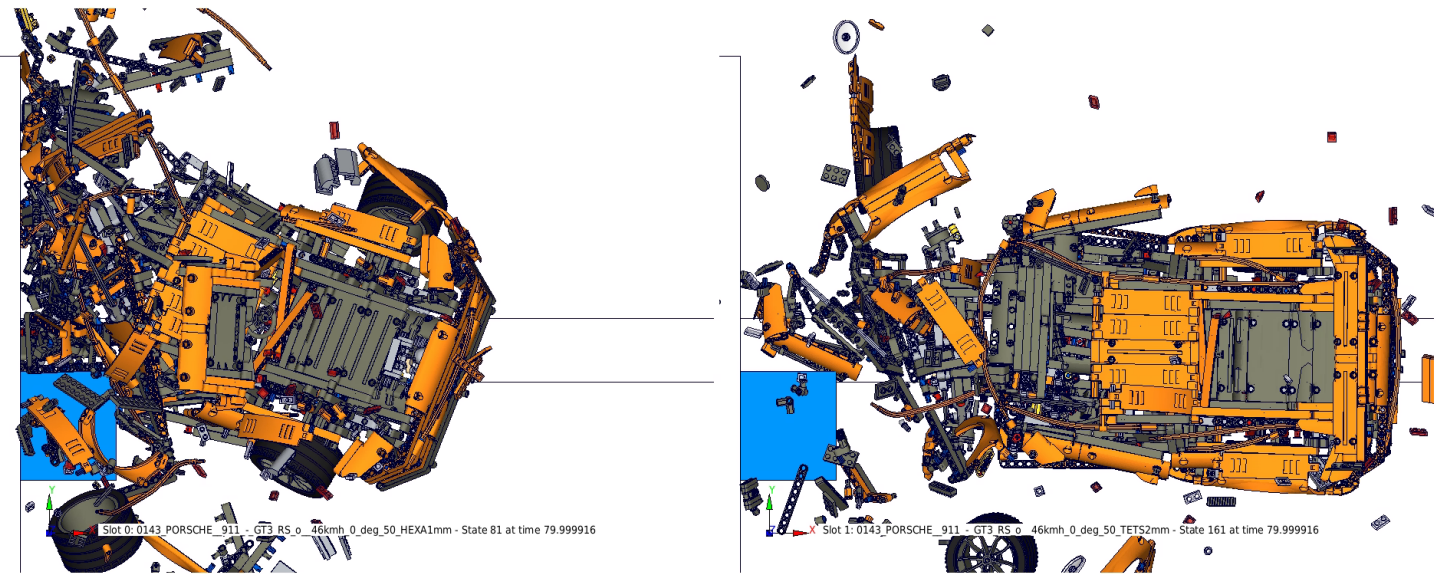


Fig. 21: 1 mm (left) and 2 mm (right) variant of the same simulation at 80 ms of the Porsche model.

As seen in Fig. 21 the 2 mm variant has not the same degree of separation of the bricks. Certainly, the reasons for this are the contact situation as well as a higher discretization error in the 2 mm variant which results in more intersections.

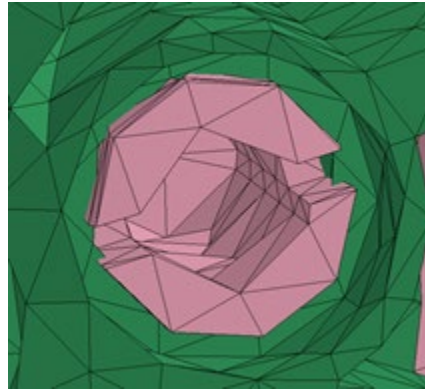


Fig. 22: Intersections between connector and technic brick due to the discretization error for the 2 mm variant.

When looking at the modelling in Fig. 22 it becomes clear that these kinds of intersections will obviously lead to nonphysical behavior of the contact between these bricks and therefore affect the way they separate.

Friction

A similar impact on model behavior may be clearly attributed to the effect of friction. Naturally, it is very hard to determine correct physical values for static and dynamic friction coefficients. So, at the beginning friction parameters of 0.1 for dynamic and 0.2 for static friction were assumed. However, by changing these values one could observe that this – as usual – has a huge impact on the model behavior. Similar to the target element size this might be because the friction directly affects how high the required forces are in order to separate bricks. A higher friction value seemingly results in higher forces needed to separate bricks and therefore results in less separation of bricks. This is illustrated in Fig. 23 where one can clearly see that lower friction results in higher degree of destruction.

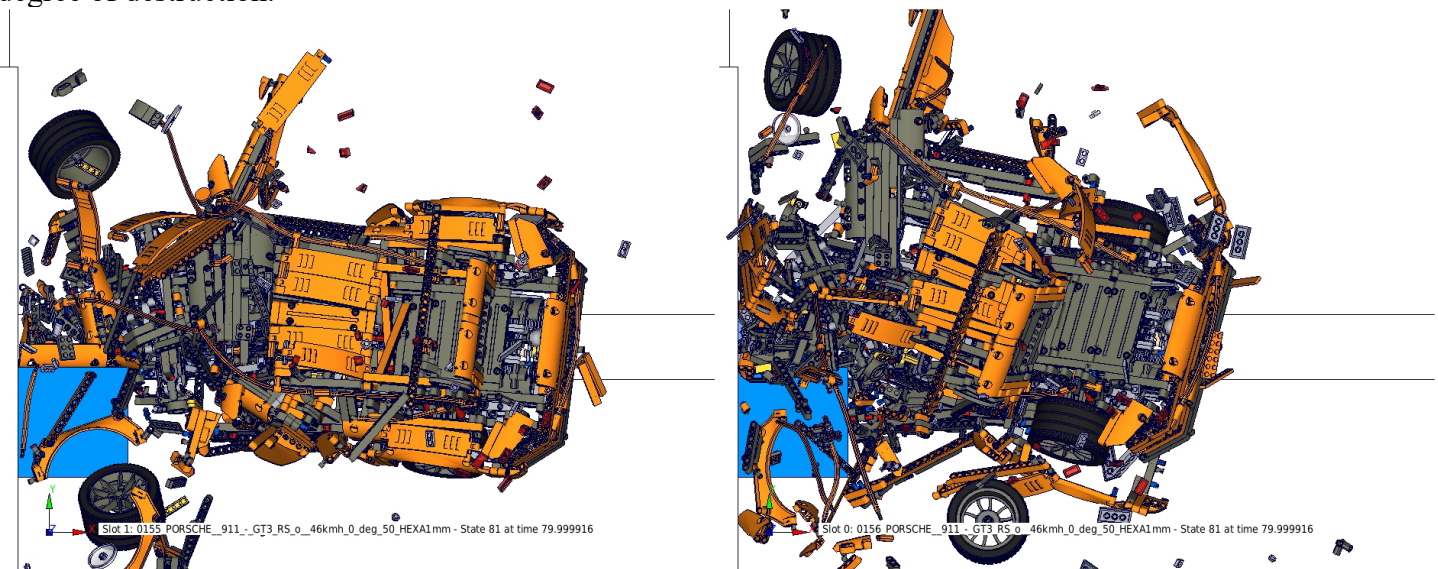


Fig. 23: Influence on model behavior for 0.05/0.1 (right) vs. 0.1/0.2 (left) dynamic/static friction coefficients at 80ms

This big impact of coefficients that are not really deterministic is problematic. However, it also gives an opportunity to counter-effect the influence of the element size and choose friction coefficients that result in an overall model behavior similar to the tests. One approach to calibrate the friction coefficients is to actually do pull out experiments with the technical bricks and reverse engineer the friction coefficients from there.

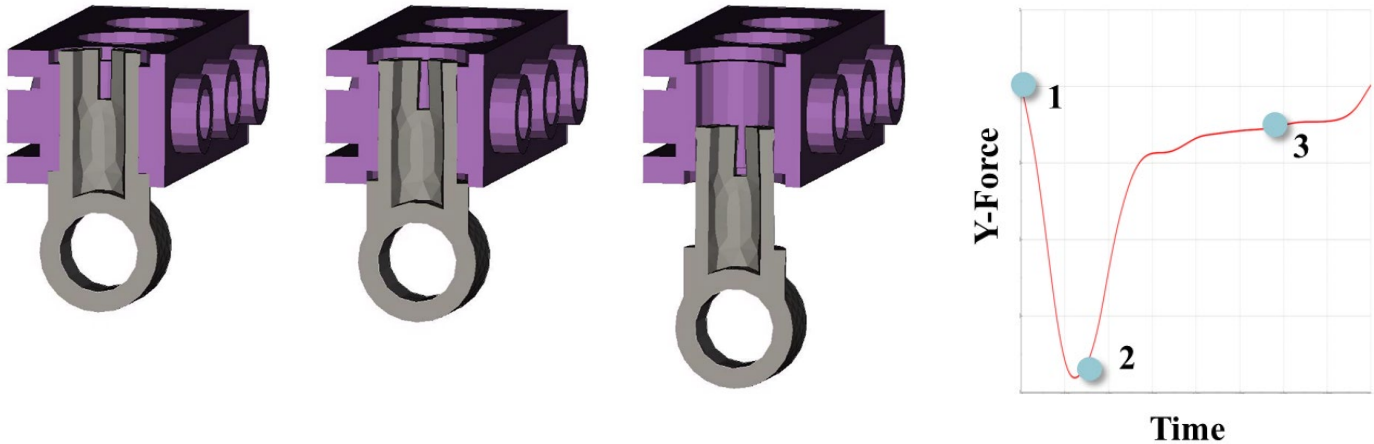


Fig. 24: Pull out experiment for determining the friction coefficients.

However, even with this validation of the friction coefficients the behavior is still not completely deterministic since there is a clear dependency of the orientation of the technical pins with regard to the shear forces needed to separate a connection. This is because the technical pins possess small slashes that allow for deformation and therefore provide the means for a controlled separation of the pins from technical bricks. However, the orientation (rotation) within the holes of the technical bricks is not deterministic and still has an effect on the separation forces. The actual magnitude of this effect has been confirmed through an experiment carried out with two simulations shown in Fig. 25, comparing a change of the rotation of all pins by 90°.

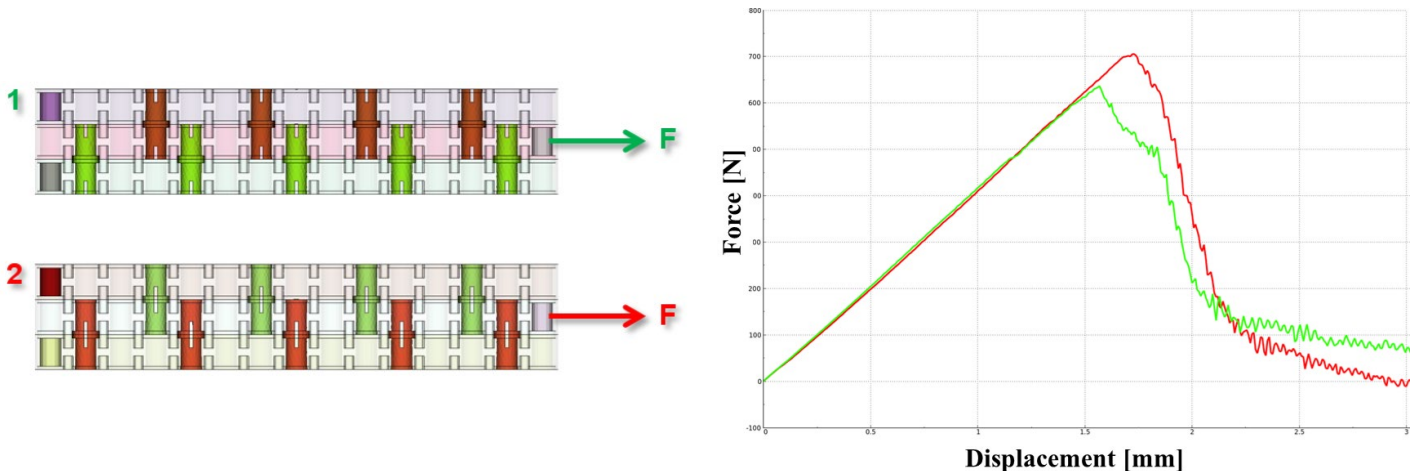


Fig. 25: Simulation to demonstrate the dependency on the orientation of technical pins

The technical pins are the most used bricks in the Porsche model and appear throughout the whole model. It is just not possible to track the orientation of each pin within the whole model. Therefore, we have to live with this small inaccuracy. Still, the fact that most of the connections are realized by these technical pins is one of the reasons that the prognosis of the crash results is as good as it was. The uncertainties when using connections with studs that depend on the correct modeling of the clamping forces are way higher.

Plastic material model

As described in section 4.2 a material card *MAT_024 was created considering actual plastic deformation of the bricks. However, as shown in Fig. 26 this static definition resulted in heavily deformed bricks.

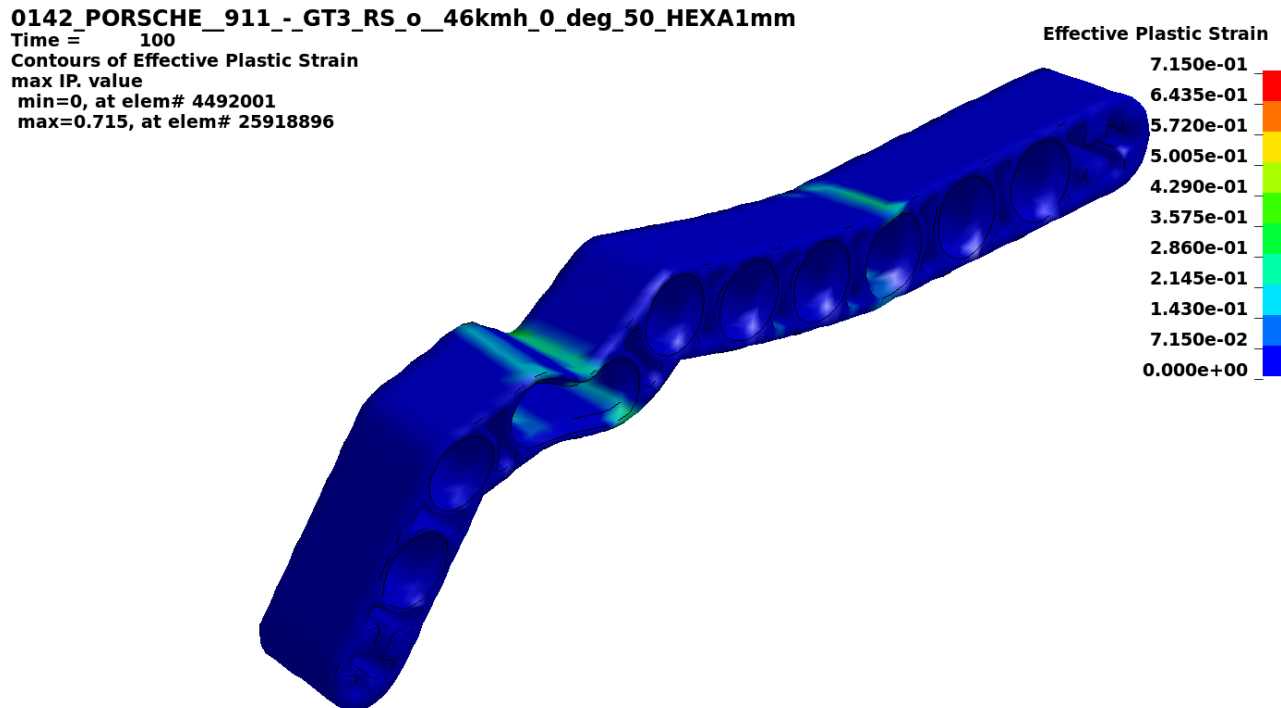


Fig. 26: Plastic deformation of brick during crash

Since this behavior was not seen in the actual physical full car test the material parameters were adjusted such that plastic behavior is not considered for the moment. One possible reason for this discrepancy is seen in the fact that clamping forces between the individual bricks were not defined yet leading to much more disintegration of the model compared to reality. Another reason is attributed to the material parameters that were calibrated for quasi static loading only. Hence any strain rate dependent effects, like work hardening under elevated load velocities are currently not taken into account. In order to capture these effects more comprehensive testing of coupons and small assemblies at higher load velocities are necessary.

Guiding blocks

In the videos, and especially in the [Making of Video](#)^[28] at YouTube, it was seen that two guiding blocks have been used in order to keep the car on track. These blocks are fixed by screws directly to bricks on the bottom floor of the car. In order to create a model of these blocks the dimensions were extracted from pictures provided by c't magazine and a [FreeCAD](#)^[29] model was created. This was in turn meshed with ANSA using shell elements. Simple rigid body material parameters were used in order to act as corresponding contact surface. The mass was added by *PART_INERTIA cards.

The guiding blocks have a considerable effect on the total rotation of the model during the crash. The front guiding block prevents the car from bouncing back and to the side and gives it some spin about the Z axis similar to what can be observed on the videos on YouTube.

Although the model behaves in many aspects differently compared to the footage from YouTube and a lot of things remain to be improved with the simulation, the results are already impressive. The overall behavior of the model is already captured to a reasonable degree with respect to the relatively little effort that was put into validation. FEA models used in real virtual product development usually undergo a lot more validation and fine tuning. This often involves many different people collaborating on material models, validation of physical parameters, calibration with the test setup and fine tuning many more details in order to capture the physical truth as good as possible.

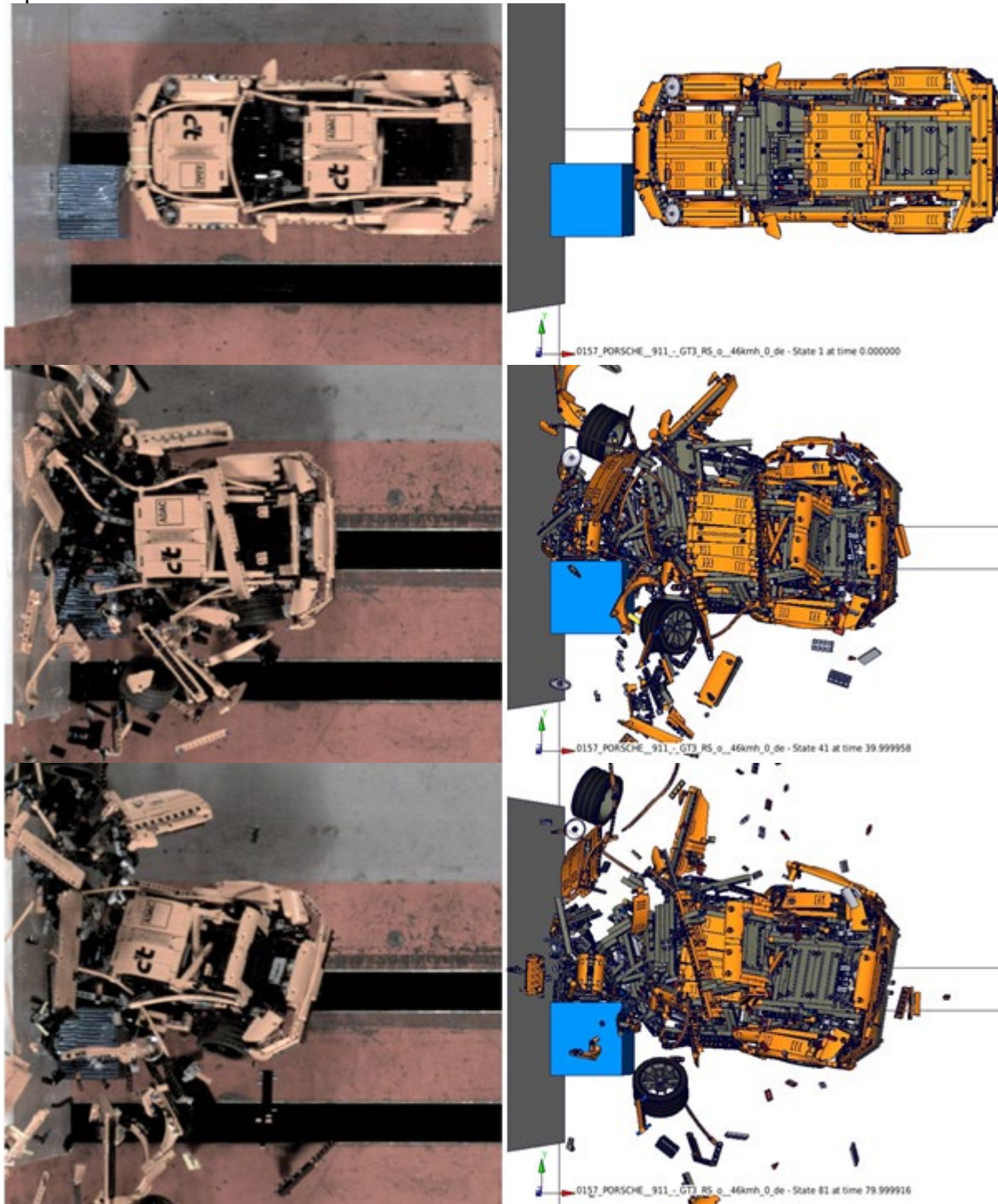


Fig. 27: Simulation results compared to images from test videos at 0, 40 and 80 ms

The Challenge “#Legowette”

When we showed the results of our first simulations of the Porsche crash to the *c't* magazine (a widely read German computer magazine) and ADAC (General German Automobile Club) who had performed the initial physical crash test in 2017, they were thrilled. However, they saw that creating a simulation of a crash with an already known outcome is the “easy thing”. The real thing would be to actually predict the outcome of a crash that has not already been performed and give a true prognosis of what will happen.

Since we already had some experience with the validated results from the crash of the Porsche model we felt confident that our prognosis would be good and we took the bet.

However, in order to make it even more challenging they also insisted that this time the crash should be a car to car crash of the Porsche model with the newer LEGO® Technic - Bugatti Chiron (42083) model. This model is even more complex and is made of 3599 individual bricks.

Furthermore, the velocity of the crash was supposed to be increased to 60km/h with one vehicle standing and one being in motion.

Setup of the Bugatti Chiron

Since the simulation process was already set up within our SDM System for the Porsche model it was relatively easy to start with the Bugatti model. The first thing was to import the CAD data into LoCo. Fortunately, the CAD data of the Bugatti^{[33][34]} was already uploaded by the same people that also prepared the CAD data of the Porsche. Once the CAD data had been imported to LoCo it became clear that only 74 bricks that were not used in the Porsche model had to be meshed. All the other meshed bricks were available from the steadily growing brick library in LoCo that can be used throughout all the LEGO® projects in our SDM System. The results of the crash of the Bugatti model with the same setup can be seen in this video on YouTube^[35]. In comparison to the Porsche model (19.5 million elements) the Bugatti FEM model contains 25 million elements. With this approach it was possible to simulate the even bigger car to car (C2C) crash of the Porsche vs. Bugatti with only little effort. The two models of the Porsche and Bugatti are set up as individual simulation projects in LoCo.

Multi-Stage-Assembly in LoCo for car to car setup

For setting up the C2C load cases we needed to bring the two cars together into one simulation. For this goal the “Multi-Stage-Assembly” feature of LoCo^[36] was used. The Multi-Stage-Assembly feature offers the possibility to import any result files/data from a simulation back into LoCo as special components and use these in the same or in other LoCo projects as input components. The special part of these components is due to their saved link to the simulation that generated them in the first place. In this manner an entire chain or tree of simulation, projects and components can be defined in LoCo. In our case the feature was used to:

- Pack all FEM files of the bricks library into one archive, and use it in the individual car pools.
- Assemble each FEM model of the two cars, the Porsche and the Bugatti, into one package after running a short datacheck simulation.
- Import each car model into a C2C pool, each as one component and just define the position and velocity parameters for them.

Therefore, a tree with two branches, each branch with two short assemblies, and subsequently the assembly of the main C2C simulation was defined for the entire C2C process (Stage 1-3 of Fig. 28). The parts of the Multi-Stage-Assembly process chain are only run again if some of the related data changes.

For example, if the data of the brick library changes, the two car models also have to be assembled again and a datacheck has to be run for each model. However, when changing parts of one of the car models only that car assembly has to be carried out again. The other car model can be used as it was within the C2C simulation without having to do another assembly or datacheck.

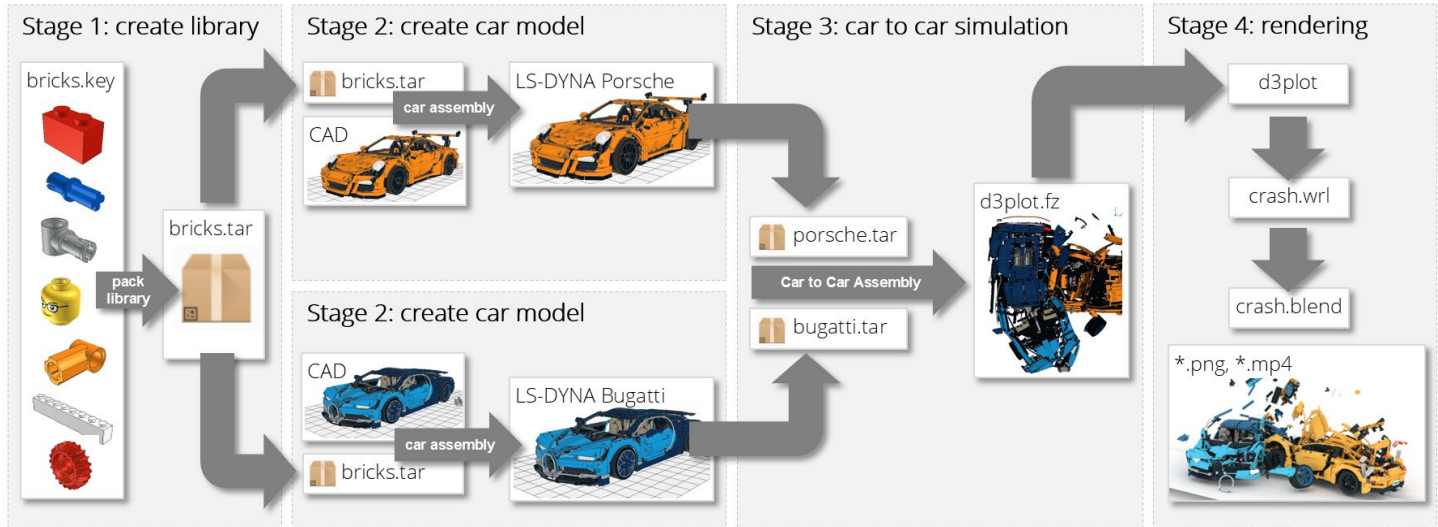


Fig. 28: Simulation process set up in SDM system with 4 stages.

This can also be used to do further processing with the actual result data from the simulations. For example, for creating photorealistic rendering of the simulations (Fig. 29) for the publication in the print magazine c't. This was achieved by using META^[23] to export each state to VRML and importing it into blender^[45] for the rendering. The whole process was automated through python scripts such that complete photorealistic videos^[46] could be created and even an VR clip^[37] has been produced that can be viewed through VR headsets.

Since the rendering stage of the Multi-Stage-Assembly accesses directly the result data from the simulations stored in the SDM system, new perspectives and rendering scenarios can be defined and automatically run for creating new light path simulations (ray tracing) without having to start new LS-DYNA simulations. The whole process is laid out in Fig. 28. This feature is needed in industrial simulation projects to set up manufacturing or process simulations where multiple simulations have to be combined.



Fig. 29: A photo realistic blender rendering of the simulated crash.

Finding the perfect crash setup

Once the process for running C2C simulations using LoCo was ready, it was trivial to create different load cases with the car models having different positions and velocities. This gave us the opportunity to try different scenarios in order to find the best crash setup for the real crash test. The goal was to find a setup for the crash that would be possible to carry out in real life at the “ADAC Technik Zentrum”^[37].

From the initial real live crashes it was known that the Porsche would sustain a velocity of at least 46km/h. However, our initial simulations with the Bugatti standing and the Porsche hitting at 46km/h were not as spectacular as the crashes where the Porsche would hit an almost rigid wall at the same speed. This is because in our new setup the energy of the crash is absorbed by two vehicles instead of one and also a part of it is consumed in the acceleration of the Bugatti model. Even though it was not clear if the small LEGO® models would sustain such a high velocity it was decided to take the risk and increase the velocity of the Porsche to 60km/h where our simulations showed a spectacular outcome.

In order to evaluate the different crash scenarios a couple of simulations had to be carried out. With about 45 million volume elements for the complete car to car setup these simulations required quite some computing time. Luckily, we were able to use the HPC system at HLRS to carry out the simulations.

For the simulation of a crash duration of 130ms this still took 54h of total calculation time for each crash scenario when utilizing 192CPUs.

In the end it was decided to conduct a side crash scenario where the Bugatti stands and the Porsche hits the Bugatti at 90° approximately at the b-pillar. (Fig. 30)

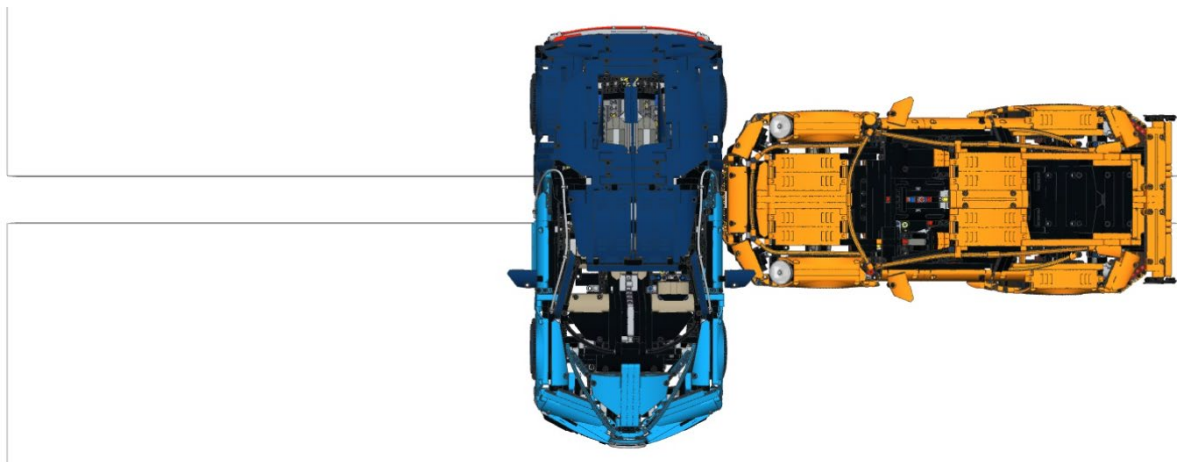


Fig. 30: Final car to car crash set up. Porsche driving at 60km/h and hitting the Bugatti on the left side.

Setup of post processing process within the SDM system

One special challenging aspect with these big models (~45 million elements) was dealing with the sparse result files. Even though we used FEMZIP^[38] to compress them, the resulting d3plot files still were about 23GB for each simulation. These had to be copied from the HPC system using scp and rsync which can take up a considerable amount of time. Furthermore, loading the sparse result files into post processing tools such as LS-PrePost^[21], Animator^[22] and META^[23] takes quite some time and the workstation required to work with the models needs a lot of RAM (64GB has proven to be a minimum required) and a powerful GPU.

Being able to do a quick review of the results without having to download the complete result files and open them in preprocessors can save valuable time. In order to demonstrate how such a setup works within an SDM system we implemented a process where the results of each simulation are post processed directly at the HPC system after the simulation and the extracted results are stored from there into the SDM system where the results appear for the user.

In such a setup the post processing on the HPC system creates videos with common perspectives as well as pictures, curves and key values. All of these can then be reviewed instantly after the simulation finishes through CAViT^[39], which is the post processing component of the SDM system.

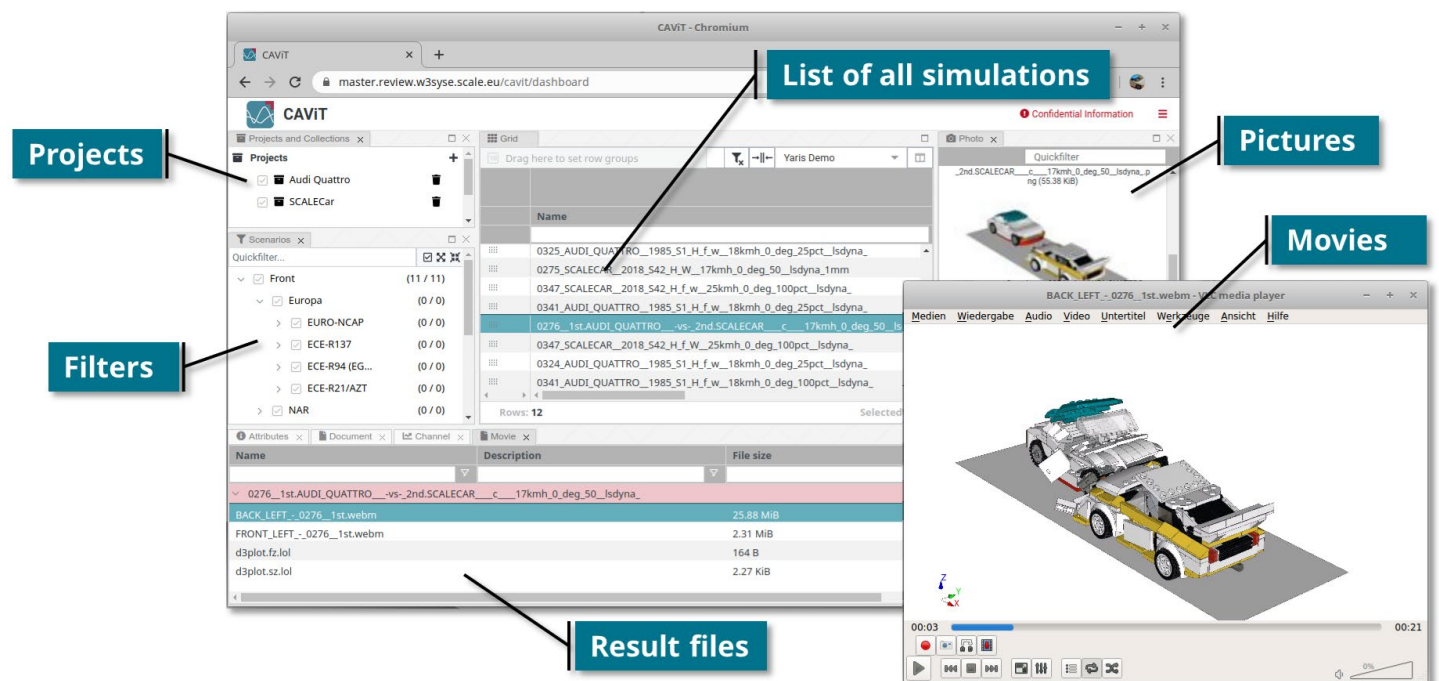


Fig. 31: Web interface for convenient accessing result data and reviewing simulations.

The results can be accessed directly from within the LoCo desktop client as well as from the CAViT web interface (Fig. 31). Accessing the results directly from the LoCo desktop client is convenient for working on workstations and downloading the sparse result data, e.g. the d3plot files. Whereas using the Web-Interface is more convenient for quick reviews, accessing reports of simulations and in circumstances where a fully fledged workstation is not available.

Being stored with the SDM system, the results are also archived and accessible for all team members working on the project independent of where they are.

The necessary steps for actually performing the extraction of data from the results is also set up and defined within the SDM system itself. For example, there are several scripts maintained within LoCo for the purpose of creating images and videos, extract curves, or perform actions during model assembly. For example, during the assembly process in LoCo which converts the positional information of the LEGO parts from the CAD data directly to LS-DYNA key words, the necessary session files for applying the correct colors to all parts are created from the color information of the CAD files automatically for the common preprocessing tools (in this case [LS-PrePost](#)^[21], [Animator](#)^[22] and [META](#)^[23]). By this high amount of automation the SDM system allows to keep cumbersome and repeating tasks away from the user and helps them focus on the real challenges of simulation instead of bothering with data handling.

Results of the Porsche vs. Bugatti crash

Before actually conducting the real crash, we gave a prognosis to c't magazin explaining in detail what we thought would be the outcome of the crash. For this purpose we met at the [HLRS](#) in Stuttgart, Germany and had the opportunity to use their [3D cave](#) for exploring the result data together with the other project participants. The whole story was covered by [c't magazine](#) in Issue 21/2019^[41]. The real crash was performed later at [ADAC Technikzentrum in Landsberg am Lech](#)^[37], Germany to validate the simulation and was covered in issue 22/2019 of [c't magazine](#)^[42]. A summary of all photos, videos and material from heise and c't magazine can also be found on their [web site](#)^[48].

Once we had the video footage of the real crash we were able to start the validation and in depth analysis of our results. For example, when we checked the first simulation results, we saw an effect that was amazing at first sight. The left front wheel of the Bugatti just stayed in place while the remaining car was flying away. That seemed wrong but it could be explained by inertia effects. The crash test conducted in Landsberg showed exactly the same behavior. This can actually be seen in [this video on YouTube](#)^[40] comparing the simulation results with the real crash (Fig. 32) as well as in detail in [this video](#)^[43] showing what's going on with the connection of the front wheel and the car.

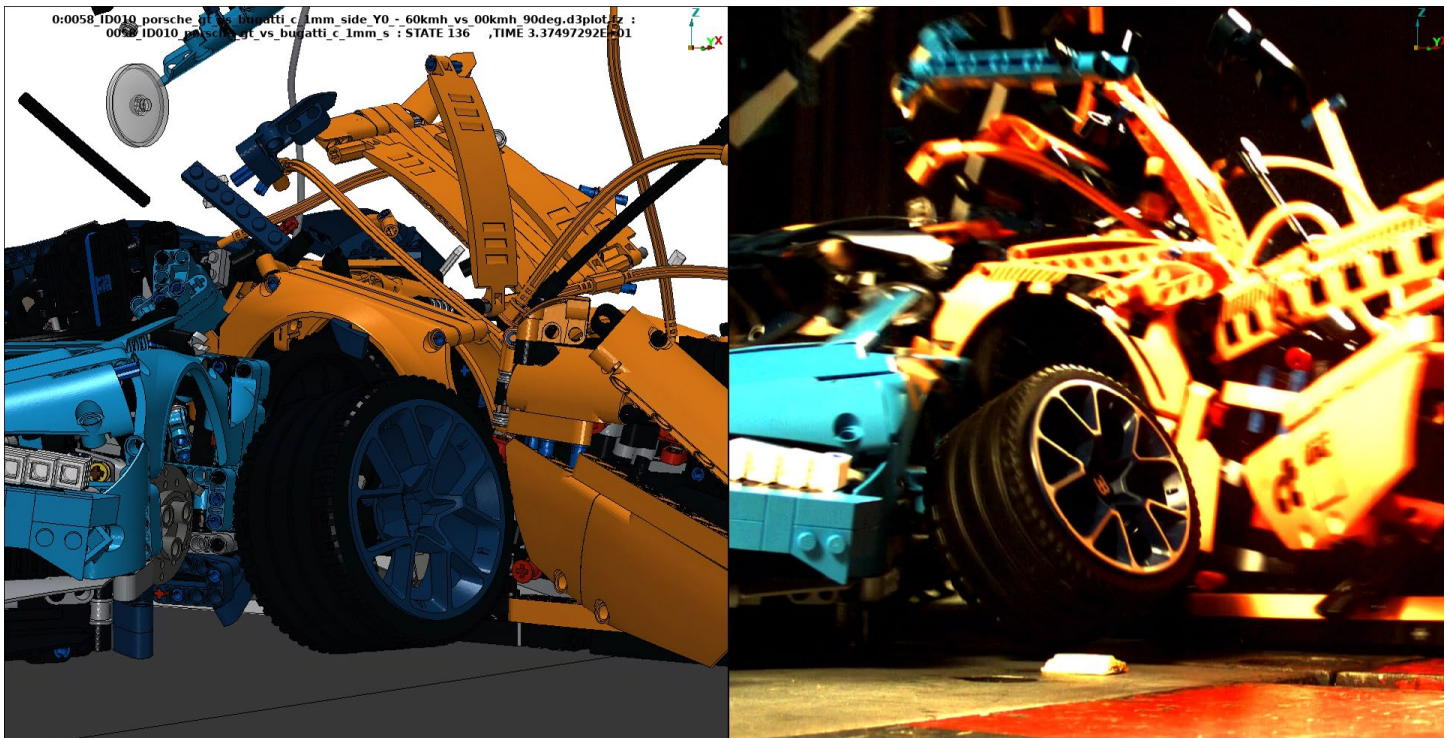


Fig. 32: Comparison of LS-DYNA simulation (left) vs. real crash footage (right) at 34ms.

We also gave some prognoses about the damage to the bricks. From the crash with the Porsche it was already known that almost no bricks got really broken so we focused on bricks with a high amount of plastic deformation. One example of this can be seen in Fig. 33. Making actual predictions about damage with the currently available material for LEGO® bricks is not reliable.

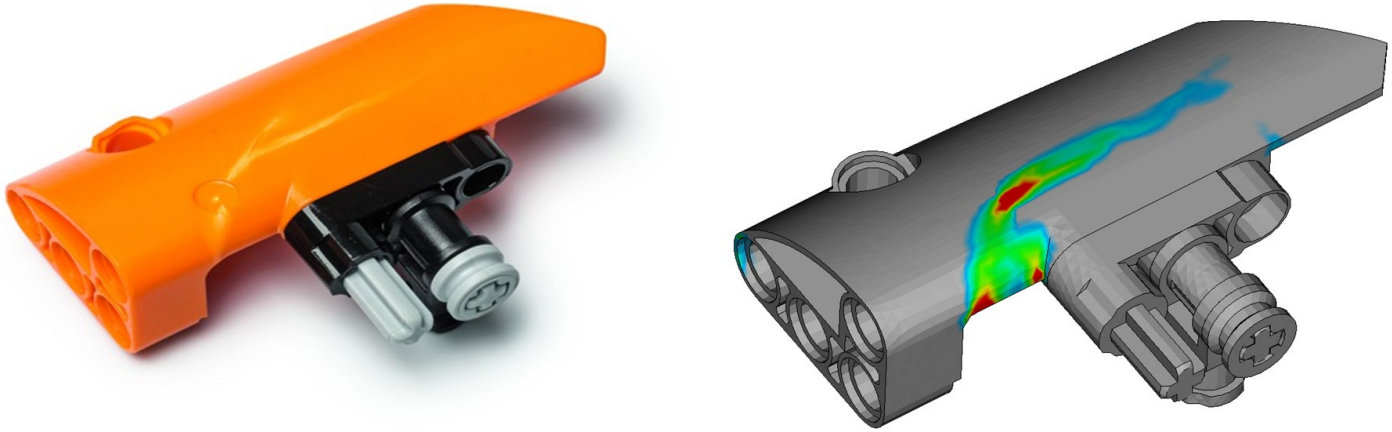


Fig. 33: Plastic deformation on real part (left) vs. simulated part (right).

The prediction of the overall model behavior during the initial crash phase was actually relatively good (Fig. 34). However, the models got a lot more disintegrated during the overall crash. This is documented in detail in the 2nd story of *c't magazine*^[42]. One reason is that the real crash was not over after the 130ms that had been simulated. Instead, the vehicles had been accelerated together to a considerable speed and smashed into the wall behind the crash zone where the two models got completely destroyed.

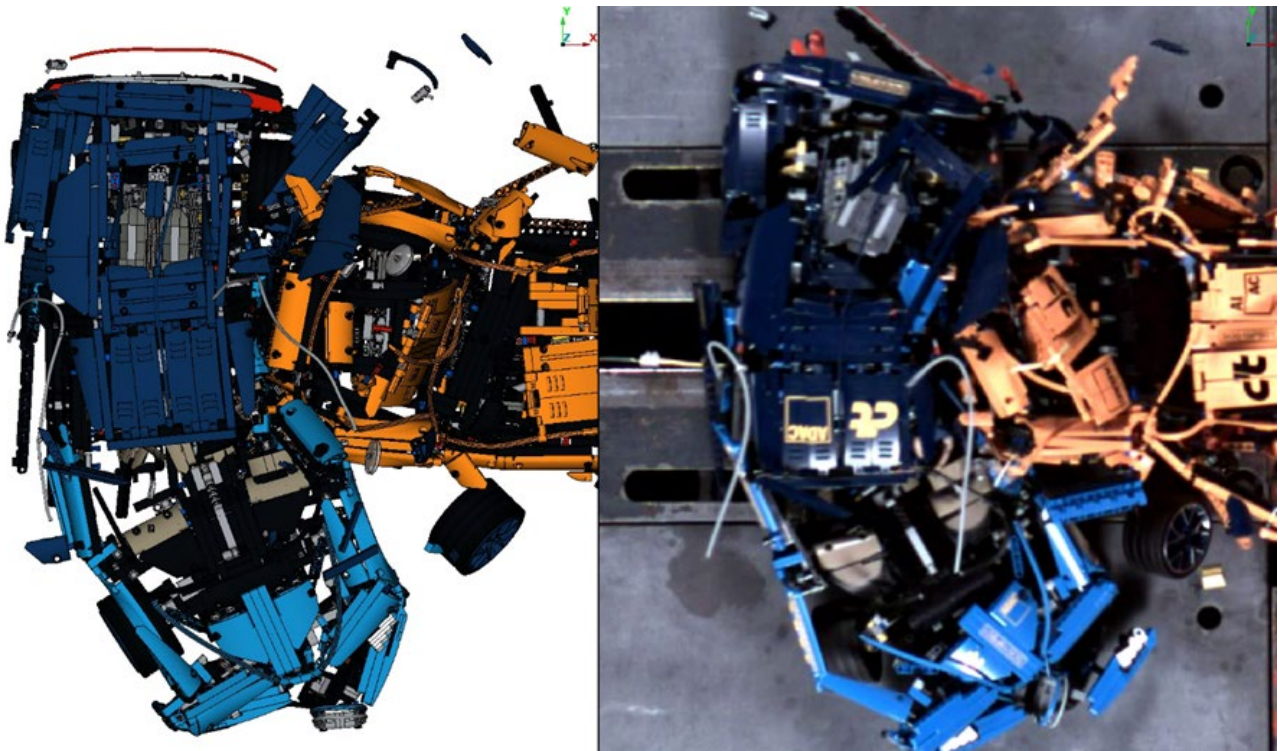


Fig. 34: Simulation (left) and real crash (right) compared at 38ms.

Nevertheless, we also underestimated the damage to the Bugatti during the crash itself. In the video of the camera that was placed in front^[44] of the crash it can clearly be seen that in the real crash the Bugatti breaks apart whereas in our simulation it stays whole. Reasons for this might be the rather simple material model or further issues related to the contact and frictions for the connections realized by technical pins.

Software development for a new Generation of SDM Products

The current SDM product offered by the SCALE GmbH is the LoCo2 rich client in combination with the CAViT rich client both being in development since 2010. These applications are showing its age given that the technological decisions are outdated, in some respects, when compared to modern standards.

For this reason, there is work ongoing to derive a new desktop client application, referred to as LoCoX, which uses the current solution as basis. With LoCoX our customers will see an update of the implementation language from Python 2 (currently being phased out) to Python 3 while at the same time replacing the user interface library from wxWidgets^[30] to Qt5^[31]. Furthermore, in preparation for a web based solution, the development team is also replacing the core components of the user interface with web based technology. Especially the part for accessing simulation results is implemented completely as a CAViT web interface but is also integrated into the desktop client LoCo.

The development of LoCo2 benefited from immediate user feedback whilst being used in the production environment of our customers. This was enabled by applying an agile release cycle. Developing LoCoX lacks this advantage, since its core features are still in active development and only a subset is deemed ready for rollout. However, feedback from workshops as well as tests from key users, albeit not fully, iron out these circumstances. While enjoying the liberty of polishing the concepts behind the different features that will be part of the final release of LoCoX, there is the risk these features do not cover all necessary functionalities for an active production environment.

For this reason, the LEGO[®] simulation efforts provide a welcome opportunity to test drive LoCoX for reliability, functionality and, especially, performance in a production environment. Performance, as mentioned in previous chapters, represents a major challenge when dealing with a high number of parts for a single model. Given that the number of parts in a LEGO[®] model exceeds those of a real car by a significant margin, demonstrating good performance for this project provides for some buffer for “real” applications.

For demonstration purposes and future training material it is planned to implement the whole process chain that is offered by our products also for the LEGO[®] crash examples. This covers every aspect from setting up a project, defining requirements, goals and limits through working with CAD data and meshing and finally assessing results in a collaborative manner.

Summary

Creating this model and simulation process was a lot of fun because of the teamwork involved. Many of the colleagues who helped in making this possible worked on the topic during their spare time or whenever there was some time in between real project work. Solving the problems associated with the process setup for the SDM system and the LS-DYNA model is quite some pleasure on its own and doing all this with an actual LEGO® model made it even more interesting and enjoyable.

The current setup with LoCo and CAViT makes it now relatively easy to set up a simulation model from LEGO® models. However, the usability of all this is not quite there yet such that young students or non FEA affine people can handle it. The final goal would be that one could download models from the OMR of Ldraw^[12], bricklink.com^[8] or create an own model, define easily some boundary conditions and run crash simulations. This will require further improvements of our Simulation Data Management System that might be achievable in the present development path of the new LoCoX desktop client and the web interface CAViT for accessing the result data. At least this will be the benchmark for the usability and ease of use for the new user interface.

The end of this journey might be that simulation has become a child's play in order to influence new generations and pass on the fascination of physics and simulation.

References

- [1] M. Thiele: "Investigation of the crash behavior of a LEGO® car ", FEA Information Engineering Solutions Volume 8, Issue 2, February 2019, pages 42-49
- [2] S. Hansen: "Lass krachen! Der Lego-Porsche-Crash bei c't", issue 12, 2017, pages 74-79
- [3] R. Bitsche, M. Thiele, T. Landschoff, M. Koch: "Recent Developments in LoCo - Instant Collaboration in Simulation Data Management", 11th European LS-DYNA Conference 2017, Salzburg, Austria
- [4] heise online, 23 May 2017, "LEGO Porsche Crashtest in Slow-Motion",
[online] Available at: <https://youtu.be/dCPWPj4JHqg>
- [5] Livermore Software Technology Corporation (1987). LS-DYNA (Version R931) [Software], <http://lstc.com/products/ls-dyna>
- [6] SCALE GmbH, LoCo [Software], <https://www.scale.eu/en/products/loco>
- [7] LDraw.org, <http://www.ldraw.org>
- [8] BrickLink Limited, <https://www.bricklink.com>
- [9] Roland Melkert (2011). LDCad (Version 1.6b) [Software], <http://www.melkert.net/LDCad>
- [10] Leonardo Zide (1998). LeoCAD (Version 18.02) [Software], <https://www.leocad.org>
- [11] BrickLink Limited (2018). Studio (Version 2.0.1_93) [Software],
<https://www.bricklink.com/v3/studio/download.page>
- [12] LDraw OMR (Official Model Repository), <http://omr.ldraw.org>
- [13] Travis Cobbs (2009). LDView (Version 4.2) [Software], <http://ldview.sourceforge.net>
- [14] The Etherpad Foundation (2008). Etherpad (Version 1.7.5) [Software], <https://etherpad.org>
- [15] Google LLC (2006). Google Docs [Software], <https://www.google.com/docs>
- [16] Collabora Ltd (2005). Collabora Online (Version 4.0.3) [Software], <https://www.collaboraoffice.com>
- [17] P. Hurbain, 2016, "42056-1 - Porsche 911 GT3 RS", CAD Data from LDraw OMR,
[online] Available at: <http://omr.ldraw.org/files?search=42056>
- [18] CCAL version 2.0, <http://creativecommons.org/licenses/by/2.0>
- [19] dk, 30 Jul. 2016, "LEGO Technic - Porsche 911 GT3 RS (42056)", CAD Data from GRABCAD,
[online] Available at: <https://grabcad.com/library/lego-technic-porsche-911-gt3-rs-42056-1>
- [20] dk, <https://grabcad.com/dk>
- [21] Livermore Software Technology Corporation (1987). LS-PrePost (Version 4.6) [Software],
<http://lstc.com/products/ls-prepost>
- [22] Gesellschaft für Numerische Simulation mbH (2004). Animator (Version 2.4.0) [Software],
<https://gns-mbh.com/products/animator>
- [23] BETA CAE Systems (1990). META (Version 15.1.0) [Software], <https://www.beta-cae.com/meta.htm>
- [24] BETA CAE Systems (1990). ANSA (Version 18.1.0) [Software], <https://www.beta-cea.com/ansa.html>

- [25] M. Thiele, 4 Dec. 2018, “LEGO Crash 25% offset barrier 17km/h: left camera compared to simulation”, [online] Available at: <https://youtu.be/fYfpCnJf0Ak>
- [26] M. Thiele, 4 Dec. 2018, “LEGO Crash 25% offset barrier 17km/h: top camera compared to simulation”, [online] Available at: <https://youtu.be/jwsiMlig1Ds>
- [27] heise online, 17 May 2017, “Lego Porsche Crashtest Teaser”, [online] Available at: <https://youtu.be/LyJ5B---Zdo>
- [28] heise online, 28 May 2017, “Lego-Crash: Making Of”, [online] Available at: <https://youtu.be/ZeBReuh4tP4?t=1100>
- [29] The FreeCAD Team (2001). FreeCAD (Version 0.18) [Software], <https://www.freecadweb.org>
- [30] Julian Smart (1992). WxWidgets (Version 3.1.2) [Software], <https://www.wxwidgets.org>
- [31] Qt Group (2014). Qt5 (Version 5.12) [Software], <https://www.qt.io>
- [32] M. Thiele: “On the Setup and Simulation of Large Scale LEGO® Models built with LS-DYNA and LoCo”, 12th European LS-DYNA Conference 2019, Koblenz, Germany
- [33] P. Hurbain, 2018, “42083-1 - Bugatti Chiron”, CAD Data from LDraw OMR, [online] Available at: <http://omr.ldraw.org/files?search=42083>
- [34] dk, 2 Sep. 2018, “LEGO Technic - Bugatti Chiron (42083)”, CAD Data from GRABCAD, [online] Available at: <https://grabcad.com/library/lego-technic-bugatti-chiron-42083-1>
- [35] M. Thiele, 21. Dec. 2019, “LEGO BUGATTI Chiron (42083) crash at 46kmh against 40% offset barrier”, [online] Available at: <https://youtu.be/GPduyH-TN8M>
- [36] A. Saharnean, M. Thiele, D. Rentsch: “LoCo - Multistage Assembly with a Wheel Generation Process Example”, 14. Deutsches LS-DYNA Forum, Oct. 2016, Bamberg [online] Available at: dynamore.de
- [37] “ADAC Technik Zentrum”, Landsberg, Germany, [online] <https://www.adac.de/sp/technikzentrum/default.aspx>
- [38] SIDACT GmbH, FEMZIP [Software], <https://www.sidact.de/femzip/>
- [39] SCALE GmbH, CAViT [Software], <https://www.scale.eu/en/products/cavit>
- [40] M. Thiele, 27. Apr. 2020, “LS-DYNA simulation vs. real LEGO® crash - Porsche (42056) vs. Bugatti (42083) CAM1”, [online] Available at: <https://youtu.be/TZ7WQq5dsdE>
- [41] S. Hansen, 27. Sep. 2019, c’t Magazin “Virtuelle Crash-Tests vs. Realität”, Issue 21/2019 [online] Available at: <https://www.heise.de/ct/artikel/Virtuelle-Crash-Tests-vs-Realitaet-4537843.html>
- [42] S. Hansen, 11. Oct. 2019, c’t Magazin “Der c’t-Legocrash 2.0: ADAC-Prüfstand versus Crash-Simulation”, Issue 22/2019 [online] Available at: <https://www.heise.de/ct/artikel/Der-c-t-Legocrash-2-0-ADAC-Pruefstand-versus-Crash-Simulation-4547127.html>
- [43] M. Thiele, 27. Apr. 2020, “LS-DYNA simulation vs. real LEGO® crash - Porsche (42056) vs. Bugatti (42083) cut left front wheel”, [online] Available at: https://youtu.be/_5WeaKYQ2IU
- [44] M. Thiele, 27. Apr. 2020, “LS-DYNA simulation vs. real LEGO® crash - Porsche (42056) vs. Bugatti (42083) cut left front wheel”, [online] Available at: <https://youtu.be/fBZK5u8wVGc>
- [45] Blender, [Software]: <https://www.blender.org/>
- [46] M. Thiele, 2. Oct. 2019, “LS-DYNA Lego Crash from SCALE/DYNAmore 2019”, [online] Available at: <https://youtu.be/3BYjrXBhUp4>
- [47] M. Thiele, 30. Sep. 2019, “Lego-Porsche-Crash als 360-Grad-3D-Video in Zeitlupe (Computersimulation)”, [online] Available at: <https://youtu.be/kESbTFekP-A>
- [48] c’t magazine, LEGO crash, <http://ct.de/crash>